



Robotics 1

Inverse kinematics

Prof. Alessandro De Luca

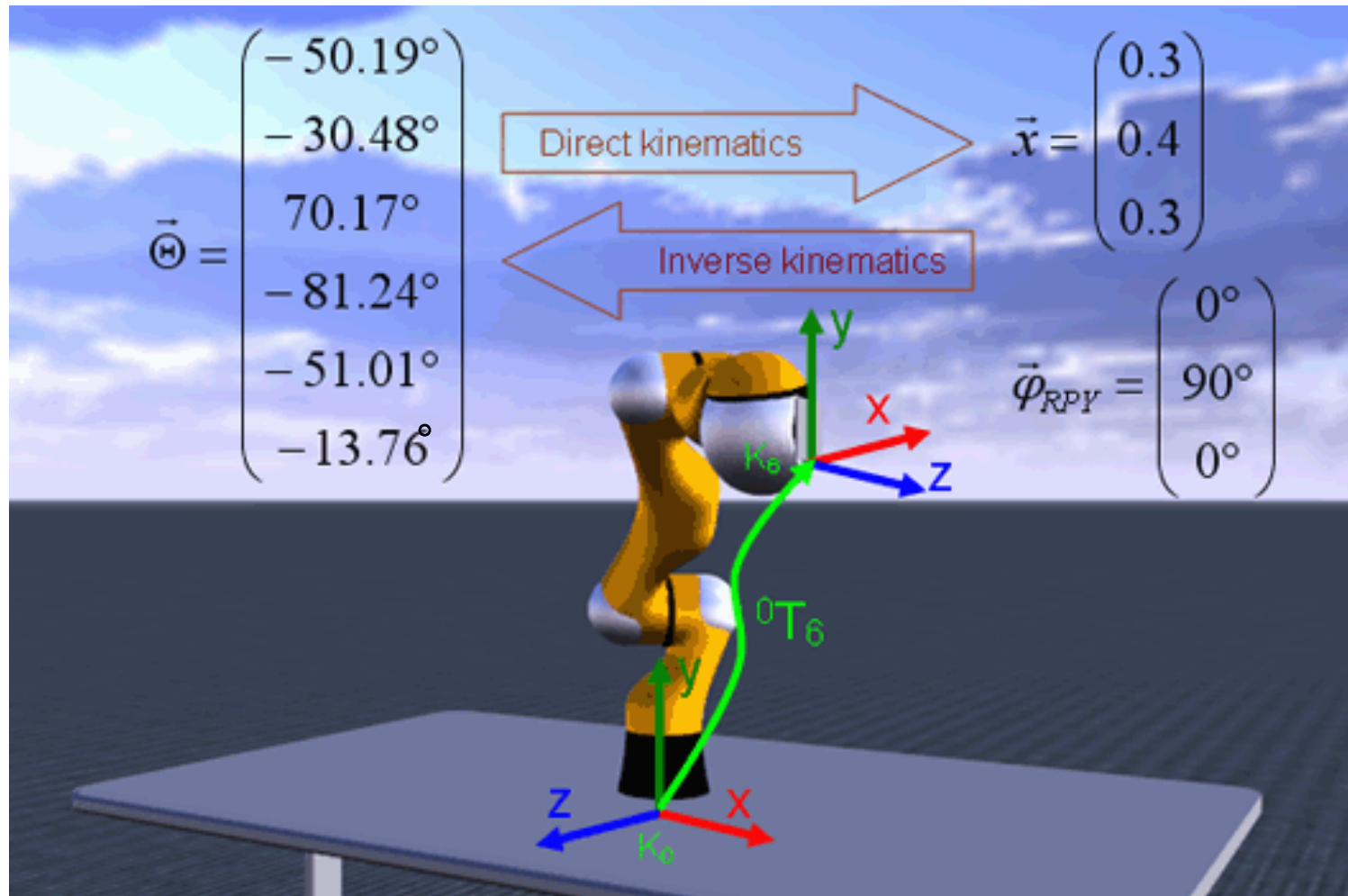
DIPARTIMENTO DI INGEGNERIA INFORMATICA
AUTOMATICA E GESTIONALE ANTONIO RUBERTI



SAPIENZA
UNIVERSITÀ DI ROMA

Inverse kinematics

what are we looking for?



direct kinematics is always unique;
how about inverse kinematics for this 6R robot?



Inverse kinematics problem

- given a desired end-effector pose (position + orientation), **find** the values of the joint variables q that will realize it
- a **synthesis** problem, with **input** data in the form

$$\blacksquare \begin{matrix} \textcolor{red}{T} \\ \textcolor{red}{r} \end{matrix} = \begin{bmatrix} R & p \\ 0^T & 1 \end{bmatrix} = {}^0A_n(q) \quad \blacksquare \textcolor{red}{r} = f_r(q), \text{ for a task function}$$

classical formulation:

inverse kinematics for a given end-effector pose $\textcolor{red}{T}$

generalized formulation:

inverse kinematics for a given value $\textcolor{red}{r}$ of task variables

- a typical **nonlinear** problem
 - **existence** of a solution (**workspace** definition)
 - uniqueness/**multiplicity** of solutions ($r \in \mathbb{R}^m, q \in \mathbb{R}^n$)
 - solution **methods**



Solvability and robot workspace

for tasks related to a desired end-effector Cartesian pose

- **primary workspace WS_1** : set of all positions p that can be reached with **at least one** orientation (ϕ or R)
 - out of WS_1 there is no solution to the problem
 - if $p \in WS_1$, there is a suitable ϕ (or R) for which a solution **exists**
- **secondary (or dexterous) workspace WS_2** : set of positions p that can be reached with **any** orientation (among those **feasible** for the robot direct kinematics)
 - if $p \in WS_2$, there exists a solution for **any** feasible ϕ (or R)
- $WS_2 \subseteq WS_1$



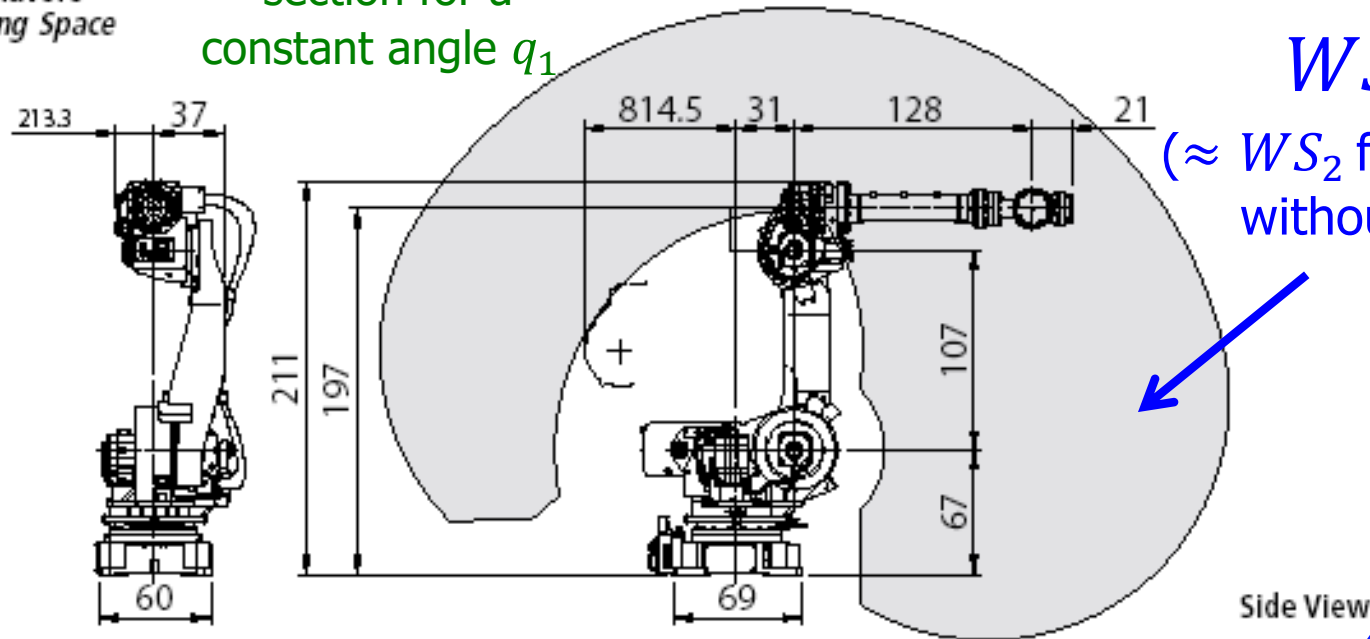
Workspace of Fanuc R-2000i/165F

Area di lavoro
Operating Space

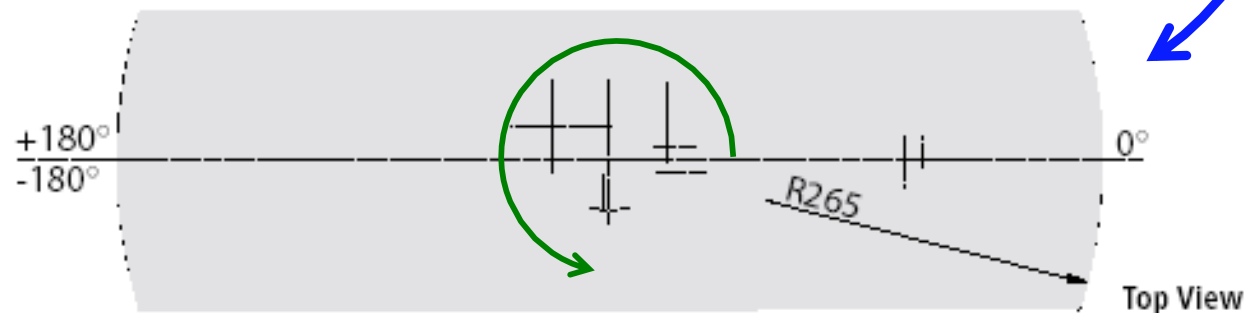
section for a
constant angle q_1

$$WS_1 \subset \mathbb{R}^3$$

($\approx WS_2$ for spherical wrist
without joint limits)



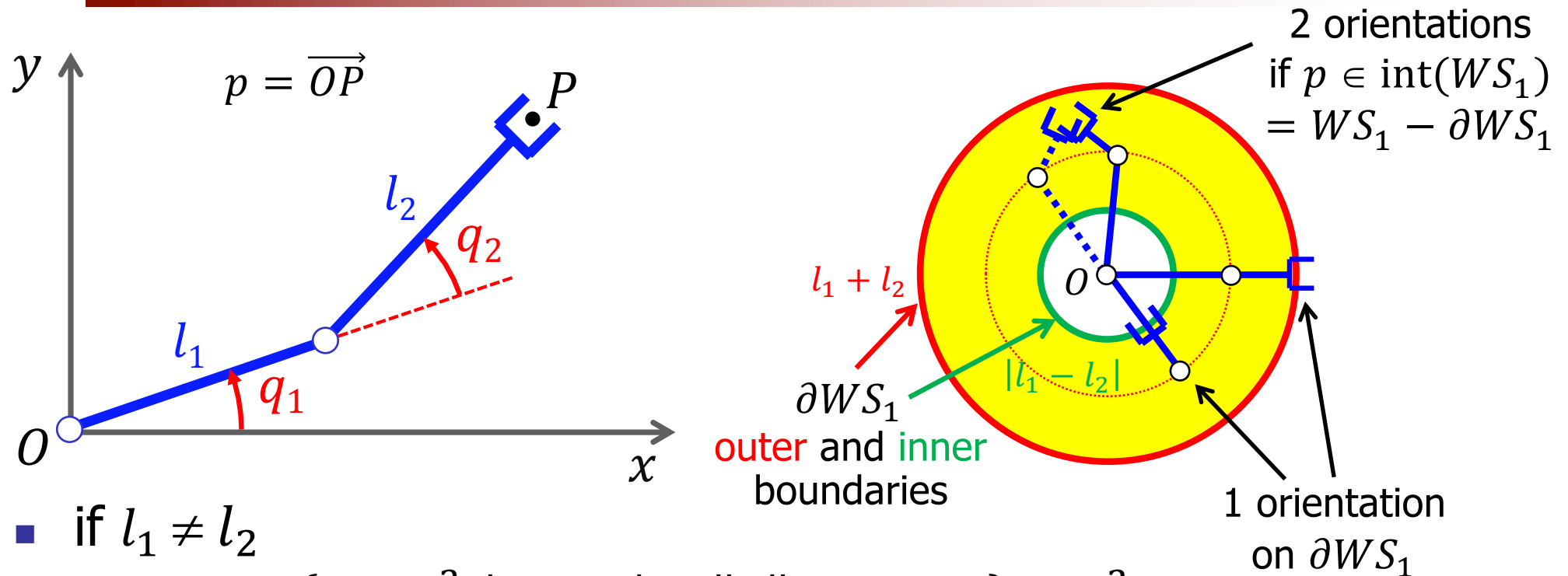
Side View



Top View

rotating the
base joint angle q_1

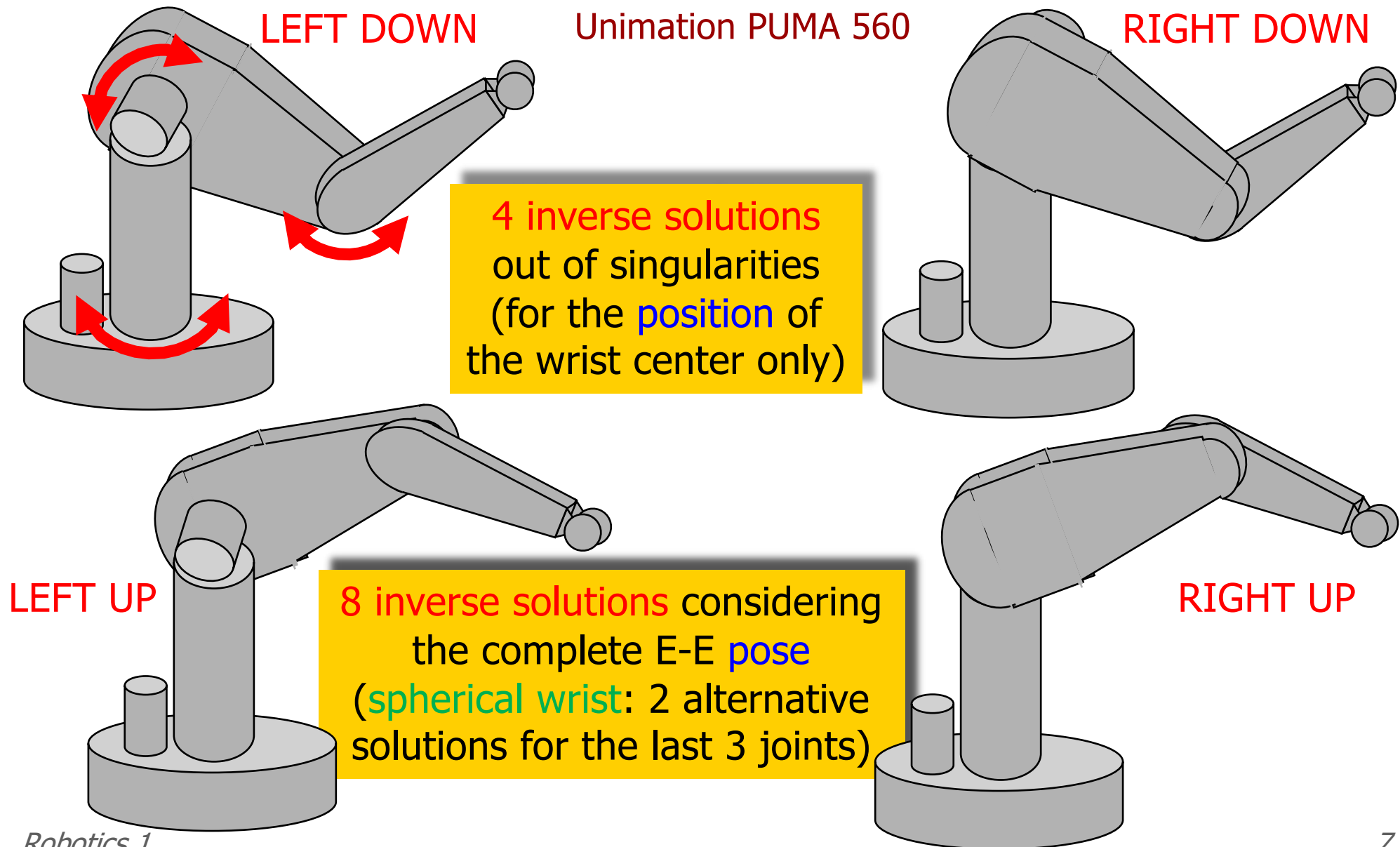
Workspace of a planar 2R arm



- if $l_1 \neq l_2$
 - $WS_1 = \{p \in \mathbb{R}^2: |l_1 - l_2| \leq \|p\| \leq l_1 + l_2\} \subset \mathbb{R}^2$
 - $WS_2 = \emptyset$
- if $l_1 = l_2 = l$
 - $WS_1 = \{p \in \mathbb{R}^2: \|p\| \leq 2l\} \subset \mathbb{R}^2$
 - $WS_2 = \{p = 0\}$ (all **feasible** orientations at the origin!... an **infinite** number)



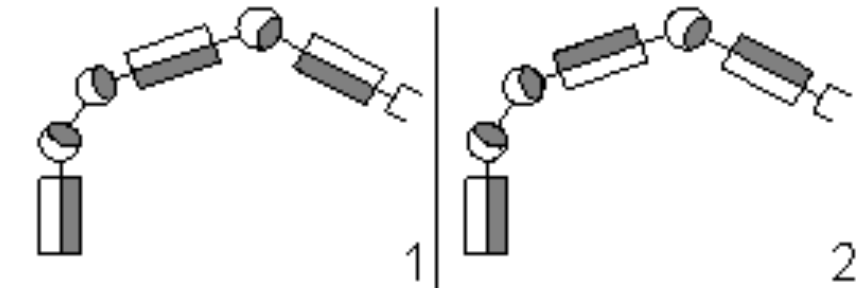
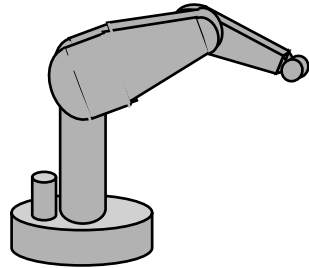
Wrist position and E-E pose inverse solutions for an articulated 6R robot



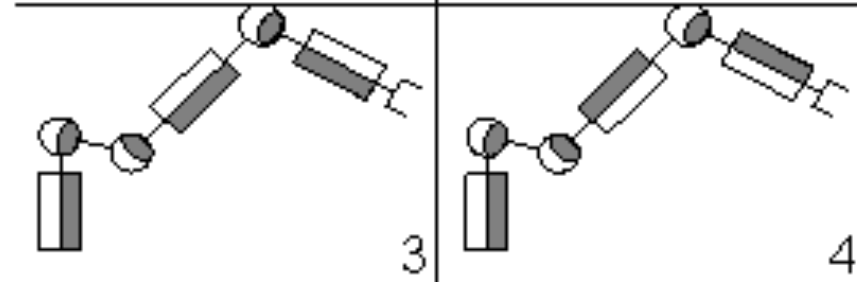
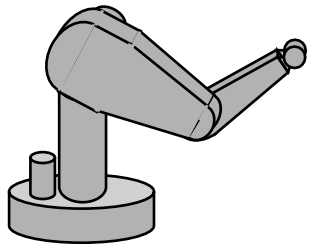
Counting/visualizing the 8 solutions of the inverse kinematics for a Unimation Puma 560



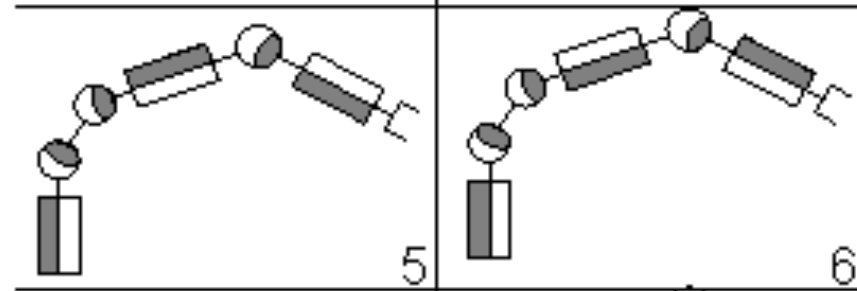
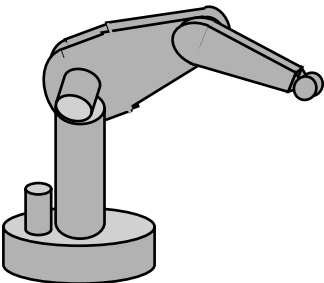
RIGHT UP



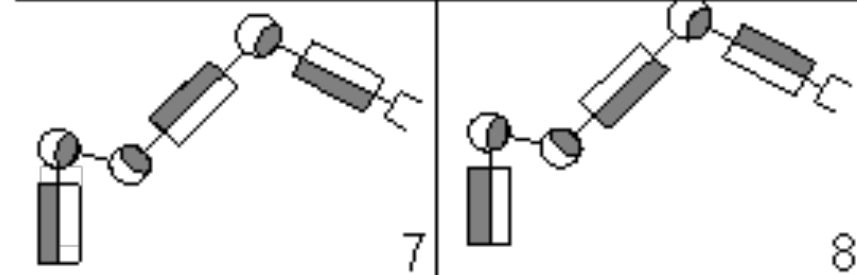
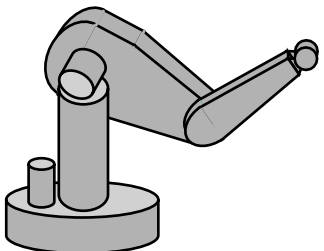
RIGHT DOWN



LEFT UP



LEFT DOWN



Inverse kinematic solutions of UR10

6-dof Universal Robot UR10, with **non-spherical** wrist



video (slow motion)

desired pose

$$p = \begin{pmatrix} -0.2373 \\ -0.0832 \\ 1.3224 \end{pmatrix} [\text{m}]$$

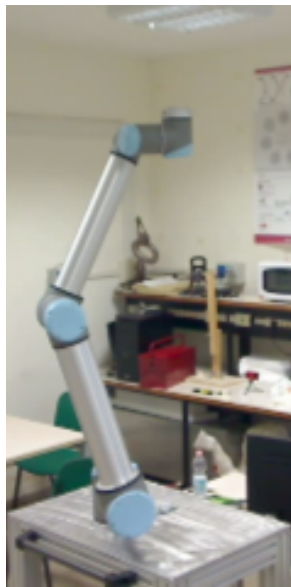
$$R = \begin{pmatrix} \sqrt{3}/2 & 0.5 & 0 \\ -0.5 & \sqrt{3}/2 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

home configuration at start

$$q = (0 \quad -\pi/2 \quad 0 \quad -\pi/2 \quad 0 \quad 0)^T [\text{rad}]$$

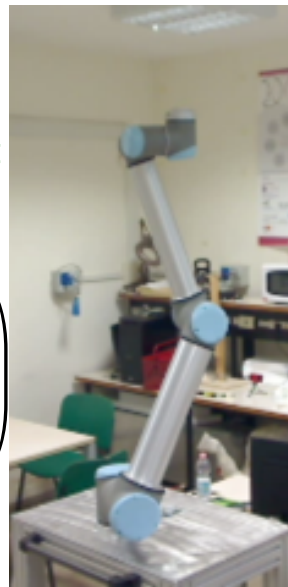


8 inverse kinematic solutions of UR10



shoulderRight
wristDown
elbowUp

$$q = \begin{pmatrix} 1.0472 \\ -1.2833 \\ -0.7376 \\ -2.6915 \\ -1.5708 \\ 3.1416 \end{pmatrix}$$



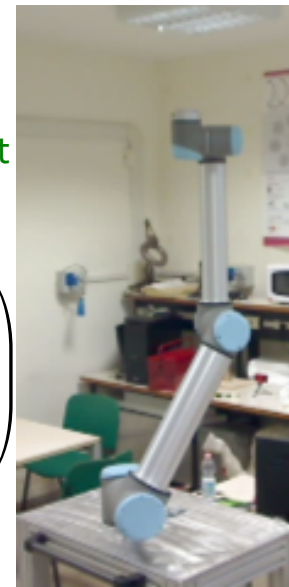
shoulderRight
wristDown
elbowDown

$$q = \begin{pmatrix} 1.0472 \\ -1.9941 \\ 0.7376 \\ 2.8273 \\ -1.5708 \\ 3.1416 \end{pmatrix}$$



shoulderRight
wristUp
elbowUp

$$q = \begin{pmatrix} 1.0472 \\ -1.5894 \\ -0.5236 \\ 0.5422 \\ 1.5708 \\ 0 \end{pmatrix}$$



shoulderRight
wristUp
elbowDown

$$q = \begin{pmatrix} 1.0472 \\ -2.0944 \\ 0.5236 \\ 0 \\ 1.5708 \\ 0 \end{pmatrix}$$



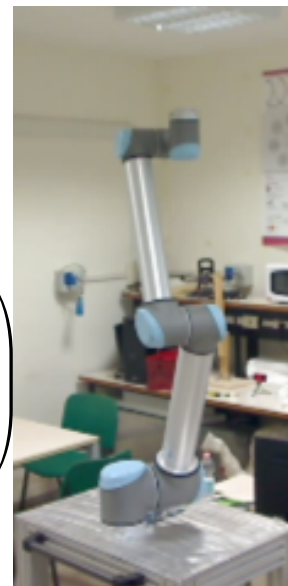
shoulderLeft
wristDown
elbowDown

$$q = \begin{pmatrix} 2.7686 \\ -1.0472 \\ -0.5236 \\ 3.1416 \\ -1.5708 \\ 1.4202 \end{pmatrix}$$



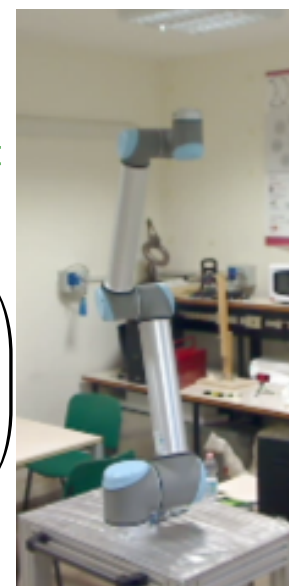
shoulderLeft
wristDown
elbowUp

$$q = \begin{pmatrix} 2.7686 \\ -1.5522 \\ 0.5236 \\ 2.5994 \\ -1.5708 \\ 1.4202 \end{pmatrix}$$



shoulderLeft
wristUp
elbowDown

$$q = \begin{pmatrix} 2.7686 \\ -1.1475 \\ -0.7376 \\ 0.3143 \\ 1.5708 \\ -1.7214 \end{pmatrix}$$



shoulderLeft
wristUp
elbowUp

$$q = \begin{pmatrix} 2.7686 \\ -1.8583 \\ 0.7376 \\ -0.4501 \\ 1.5708 \\ -1.7214 \end{pmatrix}$$



Multiplicity of solutions

few examples

- E-E positioning ($m = 2$) of a planar 2R robot
 - 2 **regular** solutions in $\text{int}(WS_1)$
 - 1 solution on ∂WS_1
 - for $l_1 = l_2$: ∞ solutions in WS_2

} **singular** solutions
- E-E positioning ($m = 3$) of an elbow-type spatial 3R robot
 - 4 **regular** solutions in WS_1 (with **singular** cases yet to be investigated ...)
- spatial 6R robot arms
 - **≤ 16 distinct solutions**, out of singularities: this “upper bound” of solutions was shown to be attained by a particular instance of “orthogonal” robot, i.e., with twist angles $\alpha_i = 0$ or $\pm\pi/2$ ($\forall i$)
 - analysis based on **algebraic transformations** of robot kinematics
 - transcendental equations are transformed into a single polynomial equation in one variable (number of roots = degree of the polynomial)
 - seek for a transformed polynomial equation of the least possible degree

Algebraic transformations

whiteboard ...



start with some **trigonometric equation** in the joint angle θ to be solved ...

$$a \sin \theta + b \cos \theta = c \quad (*)$$

introduce the algebraic transformation (... and the related inverse formulas)

$$u = \tan(\theta/2)$$

$$\Rightarrow \sin \theta = \frac{2u}{1+u^2} \quad \cos \theta = \frac{1-u^2}{1+u^2} \quad (\Rightarrow \sin^2 \theta + \cos^2 \theta = 1)$$

$$\tan \theta = \tan 2(\theta/2) = \frac{2 \tan(\theta/2)}{1 - \tan^2(\theta/2)} = \frac{2u}{1-u^2} \quad (\text{using the duplication formula})$$

substituting in **(*)**

$$a \frac{2u}{1+u^2} + b \frac{1-u^2}{1+u^2} = c \quad \Rightarrow \quad \text{polynomial equation of second degree in } u$$
$$(b+c) u^2 - 2a u - (b-c) = 0$$

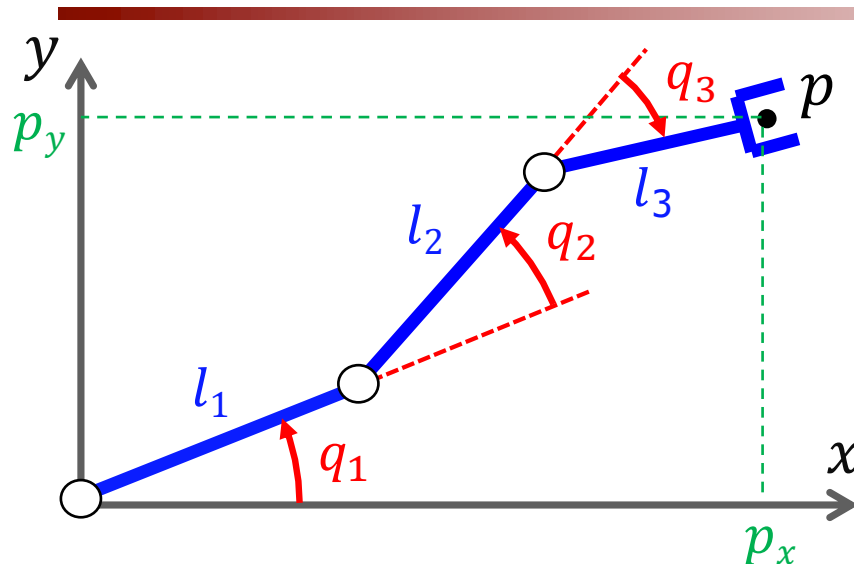
$$\Rightarrow u_{1,2} = \frac{a \pm \sqrt{a^2 + b^2 - c^2}}{b+c} \quad \Rightarrow \quad \theta_{1,2} = 2 \arctan(u_{1,2})$$

only if argument is real, else no solution



A planar 3R arm

workspace and number/type of inverse solutions



$$l_1 = l_2 = l_3 = l \quad n = 3, m = 2$$

$$WS_1 = \{p \in \mathbb{R}^2: \|p\| \leq 3l\} \subset \mathbb{R}^2$$

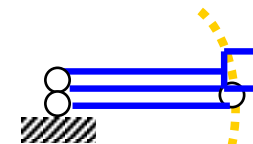
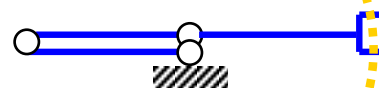
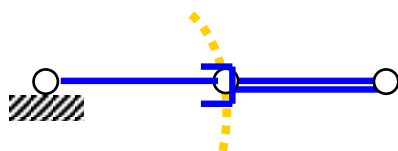
$$WS_2 = \{p \in \mathbb{R}^2: \|p\| \leq l\} \subset \mathbb{R}^2$$

any planar orientation is feasible in WS_2

1. in $\text{int}(WS_1)$, except for case 3: ∞^1 **regular** solutions, at which the E-E can take a **continuum** of ∞ orientations (but **not all** orientations in the plane!)

2. if $\|p\| = 3l$: only 1 solution, **singular**

3. if $\|p\| = l$: ∞^1 solutions, 3 of which **singular**



4. if $\|p\| < l$: ∞^1 **regular** solutions (that are **never singular**)

Workspace of a planar 3R arm with generic link lengths

$$l_{max} = \max \{l_i, i = 1, 2, 3\}$$

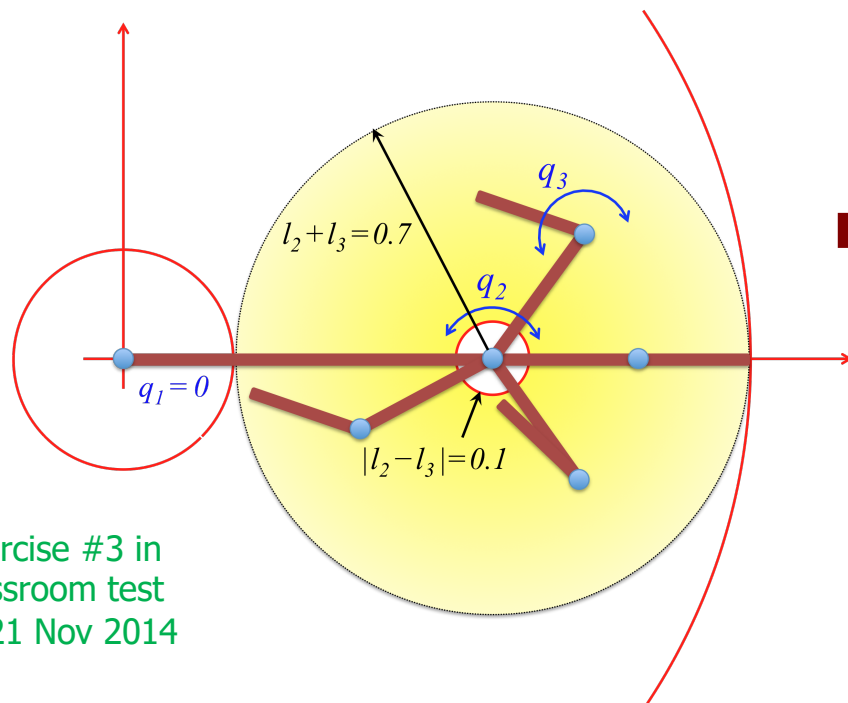
$$l_{min} = \min \{l_i, i = 1, 2, 3\}$$



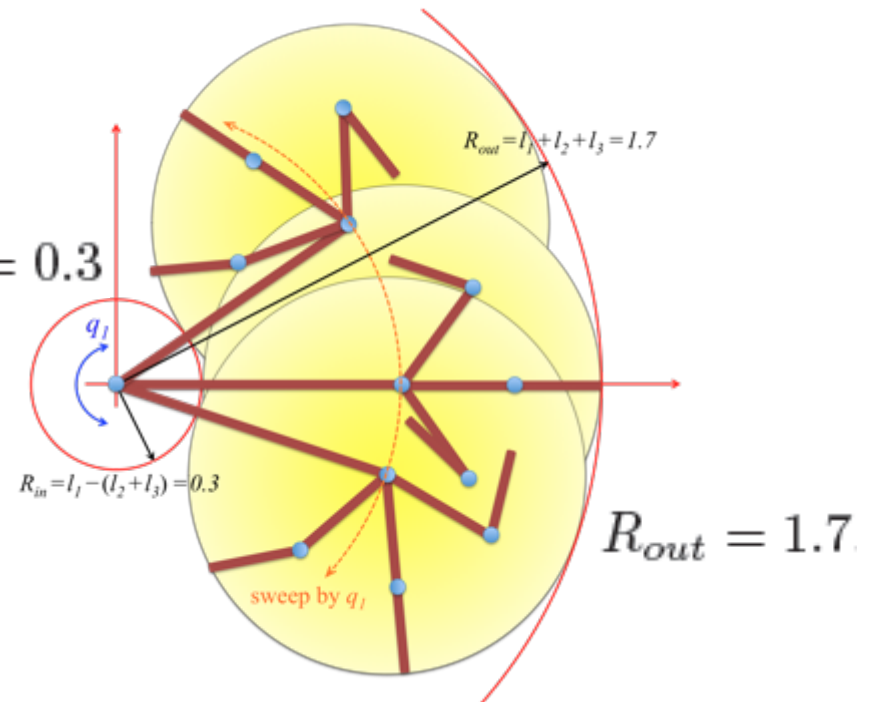
$$R_{out} = l_{min} + l_{med} + l_{max} = l_1 + l_2 + l_3$$

$$R_{in} = \max \{0, l_{max} - (l_{med} + l_{min})\}$$

a) $l_1 = 1, l_2 = 0.4, l_3 = 0.3$ [m] $\Rightarrow l_{max} = l_1 = 1, l_{med} = l_2 = 0.4, l_{min} = l_3 = 0.3$



$$R_{in} = 0.3$$



Exercise #3 in
classroom test
of 21 Nov 2014

b) $l_1 = 0.5, l_2 = 0.7, l_3 = 0.5$ [m] $\Rightarrow l_{max} = l_2 = 0.7, l_{med} = l_{min} = l_1(\text{or } l_3) = 0.5$



$$R_{in} = 0, R_{out} = 1.7$$



Multiplicity of solutions

summary of the general cases

- if $m = n$
 - $\nexists$ solutions
 - a finite number of solutions (regular/generic case)
 - “degenerate” solutions: infinite or finite set, but anyway different in number from the generic case (singularity)
- if $m < n$ (robot is kinematically redundant for the task)
 - $\nexists$ solutions
 - ∞^{n-m} solutions (regular/generic case)
 - a finite or infinite number of singular solutions
- use of the term singularity will become clearer when dealing with differential kinematics
 - instantaneous velocity mapping from joint to task velocity
 - lack of full rank of the associated $m \times n$ Jacobian matrix $J(q)$

Dexter 8R robot arm

- $m = 6$ (position and orientation of E-E)
- $n = 8$ (all revolute joints)
- ∞^2 inverse kinematic solutions (redundancy degree = $n - m = 2$)

video

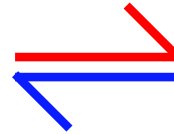


exploring inverse kinematic solutions by a robot self-motion



Solution methods

ANALYTICAL solution
(in closed form)



NUMERICAL solution
(in iterative form)

- preferred, if it can be found*
- use ad-hoc geometric inspection
- algebraic methods (solution of polynomial equations)
- systematic ways for generating a reduced set of equations to be solved

* **sufficient conditions for 6-dof arms**

- 3 consecutive rotational joint axes are incident (e.g., spherical wrist), **or**
- 3 consecutive rotational joint axes are parallel

D. Pieper, PhD thesis, Stanford University, 1968

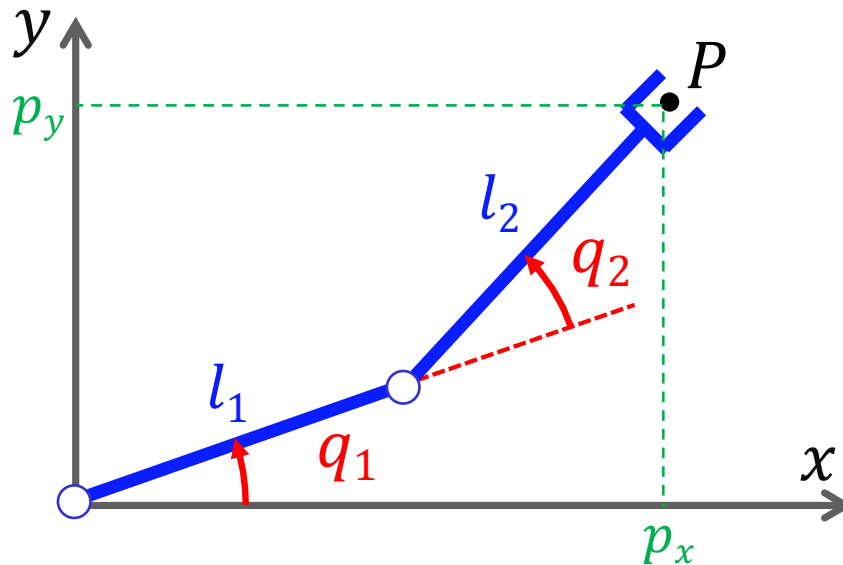
- certainly needed if $n > m$ (redundant case) or at/close to singularities
- slower, but easier to be set up
- in its basic form, it uses the (analytical) **Jacobian matrix** of the direct kinematics map

$$J_r(q) = \frac{\partial f_r(q)}{\partial q}$$

- **Newton** method, **Gradient** method, and so on...



Inverse kinematics of planar 2R arm



direct kinematics

$$p_x = l_1 c_1 + l_2 c_{12}$$

$$p_y = l_1 s_1 + l_2 s_{12}$$

$\underbrace{\quad}_{\text{data}}$ q_1, q_2 unknowns

“squaring and summing” the equations of the direct kinematics

$$p_x^2 + p_y^2 - (l_1^2 + l_2^2) = 2l_1l_2(c_1c_{12} + s_1s_{12}) = 2l_1l_2c_2$$

and from this

$$c_2 = (p_x^2 + p_y^2 - (l_1^2 + l_2^2)) / 2l_1l_2, \quad s_2 = \pm \sqrt{1 - c_2^2}$$

↑
must be in $[-1, 1]$ (else, point P
is outside robot workspace!)

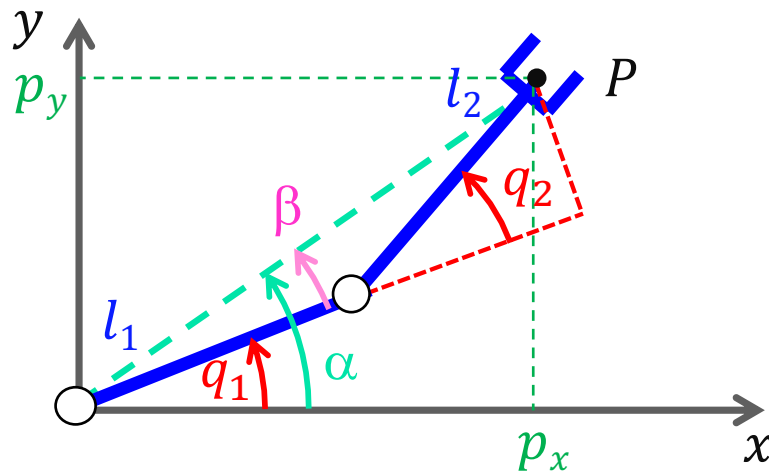
↑
2 solutions

$$q_2 = \text{atan2}\{s_2, c_2\}$$

↑
in analytical form



Inverse kinematics of 2R arm (cont'd)



by geometric inspection

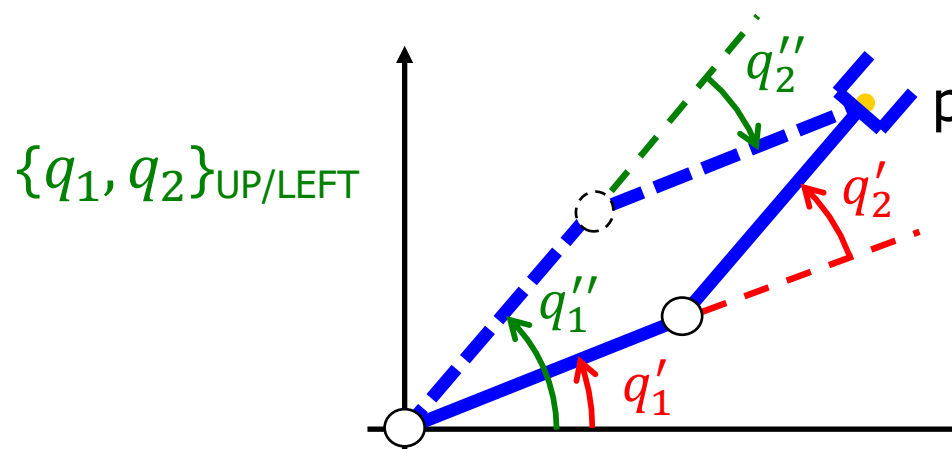
$$q_1 = \alpha - \beta$$



2 solutions
(one for each value of s_2)

$$q_1 = \text{atan2}\{p_y, p_x\} - \text{atan2}\{l_2 s_2, l_1 + l_2 c_2\}$$

note: difference of atan2's needs to be re-expressed in $(-\pi, \pi]$!



$\{q_1, q_2\}_{\text{DOWN/RIGHT}}$

q_2' and q_2'' have same absolute value, but opposite signs



Algebraic solution for q_1

another
solution
method...

$$\left. \begin{aligned} p_x &= l_1 c_1 + l_2 c_{12} = l_1 c_1 + l_2 (c_1 c_2 - s_1 s_2) \\ p_y &= l_1 s_1 + l_2 s_{12} = l_1 s_1 + l_2 (s_1 c_2 + c_1 s_2) \end{aligned} \right\} \text{linear in } s_1 \text{ and } c_1$$

$$\underbrace{\begin{bmatrix} l_1 + l_2 c_2 & -l_2 s_2 \\ l_2 s_2 & l_1 + l_2 c_2 \end{bmatrix}} \begin{bmatrix} c_1 \\ s_1 \end{bmatrix} = \begin{bmatrix} p_x \\ p_y \end{bmatrix}$$

$$\det = l_1^2 + l_2^2 + 2l_1 l_2 c_2 > 0$$

except if $l_1 = l_2$ and $c_2 = -1$
being then q_1 undefined
(singular case: ∞^1 solutions)

$$q_1 = \text{atan2}\{s_1, c_1\}$$

$$= \text{atan2}\left\{\frac{p_y(l_1 + l_2 c_2) - p_x l_2 s_2}{\det}, \frac{p_x(l_1 + l_2 c_2) + p_y l_2 s_2}{\det}\right\}$$

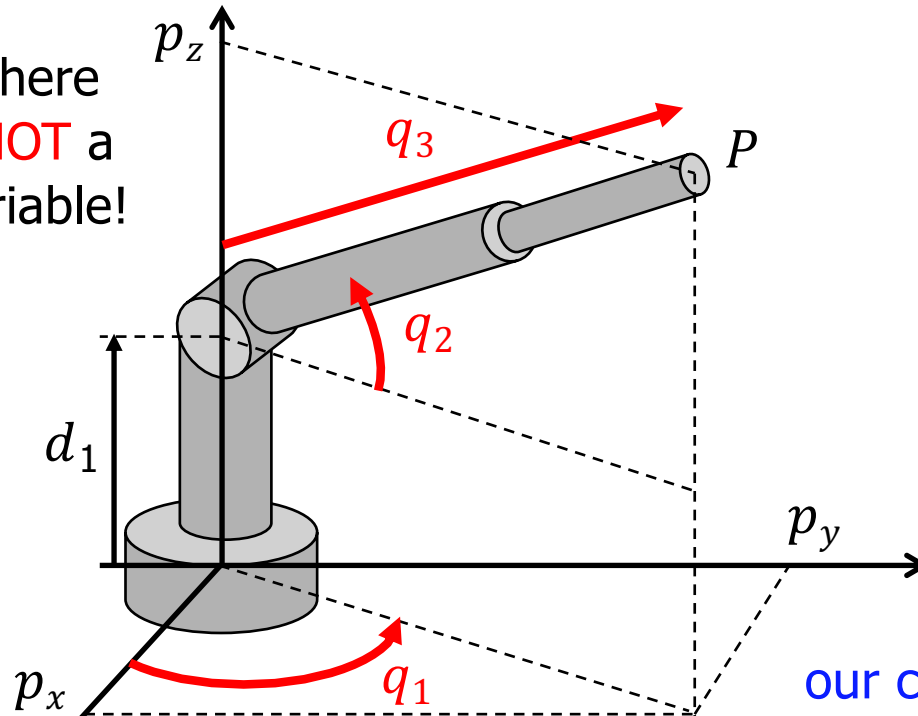
notes: a) this method provides directly the result in $(-\pi, \pi]$

b) when evaluating atan2, $\det > 0$ can be in fact eliminated
from the expressions of s_1 and c_1 (not changing the result)



Inverse kinematics of polar (RRP) arm

note: here q_2 is **NOT** a DH variable!



direct kinematics

$$p_x = q_3 c_2 c_1$$

$$p_y = q_3 c_2 s_1$$

$$p_z = d_1 + q_3 s_2$$

$$p_x^2 + p_y^2 + (p_z - d_1)^2 = q_3^2$$

$$q_3 = + \sqrt{p_x^2 + p_y^2 + (p_z - d_1)^2}$$

our choice: take here only the positive value...

if $q_3 = 0$, then q_1 and q_2 remain both undefined (stop); **else**

$$q_2 = \text{atan2} \left\{ (p_z - d_1) / q_3, \pm \sqrt{p_x^2 + p_y^2} / q_3 \right\}$$

(if we stop, it is a **singular** case:
 ∞^2 or ∞^1
solutions)

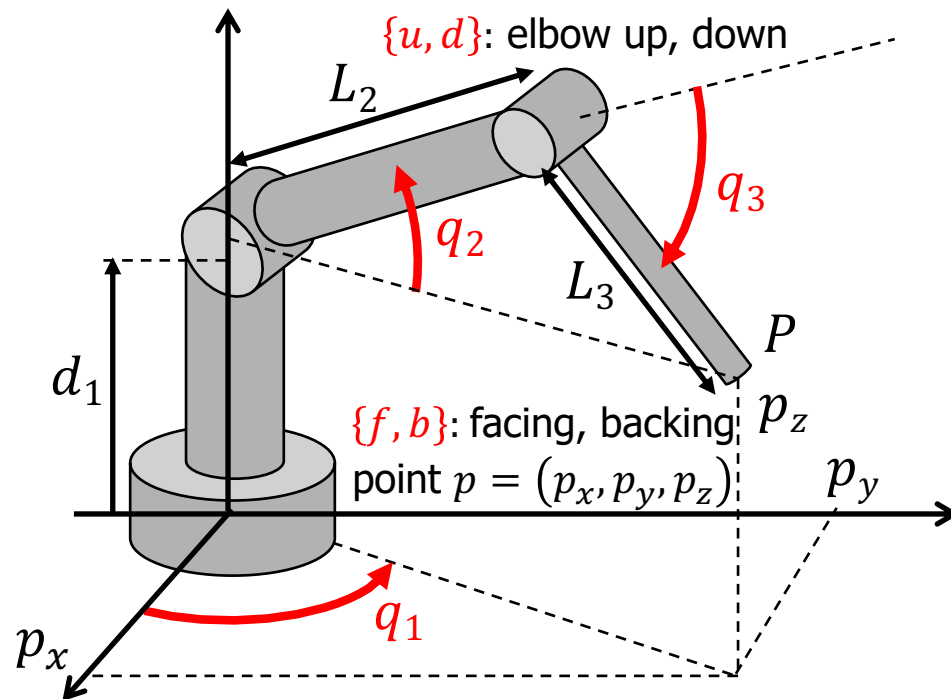
if $p_x^2 + p_y^2 = 0$, then q_1 remains undefined (stop); **else**

$$q_1 = \text{atan2} \{ p_y / c_2, p_x / c_2 \}$$

(2 **regular** solutions $\{q_1, q_2, q_3\}$)

eliminating $q_3 > 0$ from both arguments

Inverse kinematics of 3R elbow-type arm



symmetric structure **without** offsets
e.g., first 3 joints of Mitsubishi PA10 robot

$WS_1 = \{\text{spherical shell centered at } (0,0,d_1),$
with outer radius $R_{out} = L_2 + L_3$
and inner radius $R_{in} = |L_2 - L_3|\}$

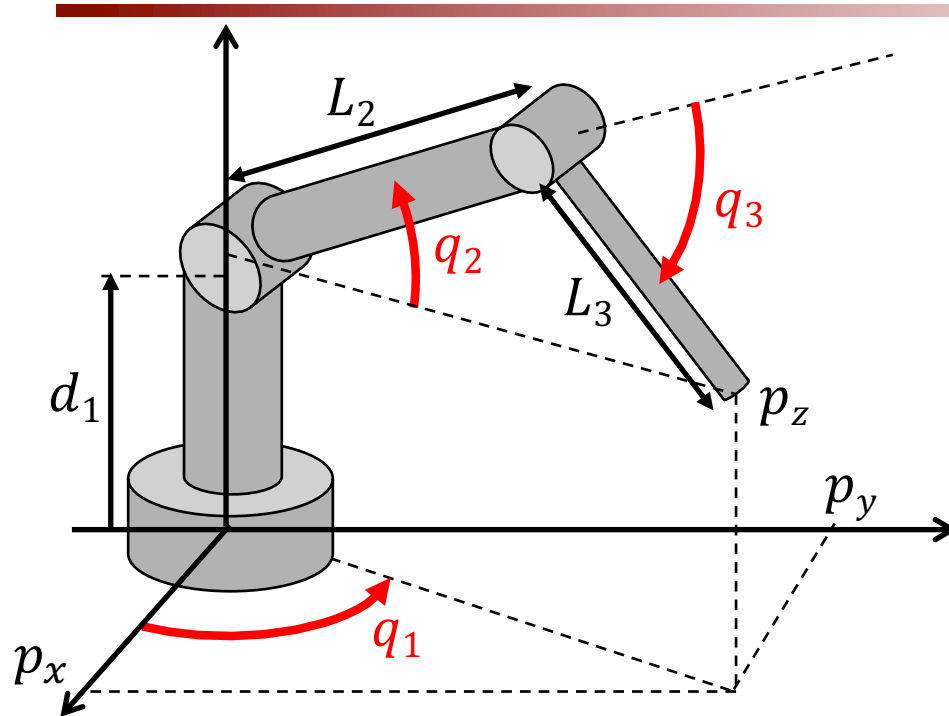


4 **regular** inverse
kinematics solutions in WS_1

more details (e.g., full handling of **singular cases**)
can be found in the solution of Exercise #1
in written exam of 11 Apr 2017

Inverse kinematics of 3R elbow-type arm

step 1



direct
kinematics

$$\begin{aligned} p_x &= c_1(L_2c_2 + L_3c_{23}) \\ p_y &= s_1(L_2c_2 + L_3c_{23}) \\ p_z &= d_1 + L_2s_2 + L_3s_{23} \end{aligned}$$

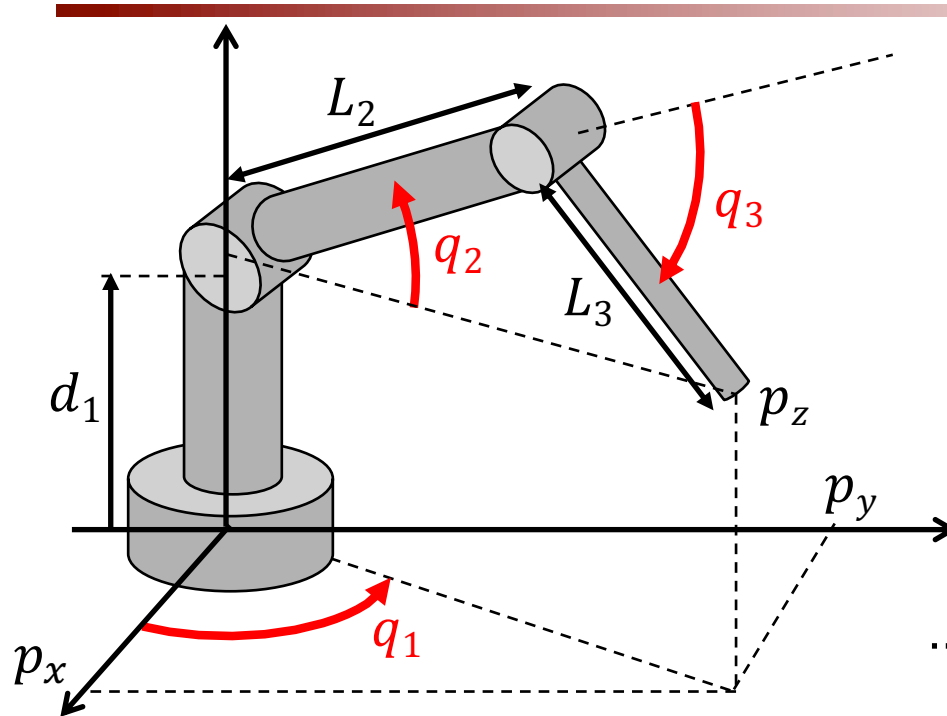
$$\begin{aligned} p_x^2 + p_y^2 + (p_z - d_1)^2 &= c_1^2(L_2c_2 + L_3c_{23})^2 + s_1^2(L_2c_2 + L_3c_{23})^2 + (L_2s_2 + L_3s_{23})^2 \\ &= \dots = L_2^2 + L_3^2 + 2L_2L_3(c_2c_{23} + s_2s_{23}) = L_2^2 + L_3^2 + 2L_2L_3c_3 \end{aligned}$$

$$c_3 = (p_x^2 + p_y^2 + (p_z - d_1)^2 - L_2^2 - L_3^2) / 2L_2L_3 \in [-1, +1] \text{ (else, } p \text{ is out of workspace!)}$$

$$\downarrow$$

$$\pm s_3 = \pm \sqrt{1 - c_3^2} \quad \Rightarrow \quad \text{two solutions} \quad \left\{ \begin{array}{l} q_3^{(+)} = \text{atan2}\{s_3, c_3\} \\ q_3^{(-)} = \text{atan2}\{-s_3, c_3\} = -q_3^{(+)} \end{array} \right.$$

Inverse kinematics of 3R elbow-type arm step 2



direct
kinematics

$$\begin{aligned} p_x &= c_1(L_2c_2 + L_3c_{23}) \\ p_y &= s_1(L_2c_2 + L_3c_{23}) \\ p_z &= d_1 + L_2s_2 + L_3s_{23} \end{aligned}$$

... being $p_x^2 + p_y^2 = (L_2c_2 + L_3c_{23})^2 > 0$

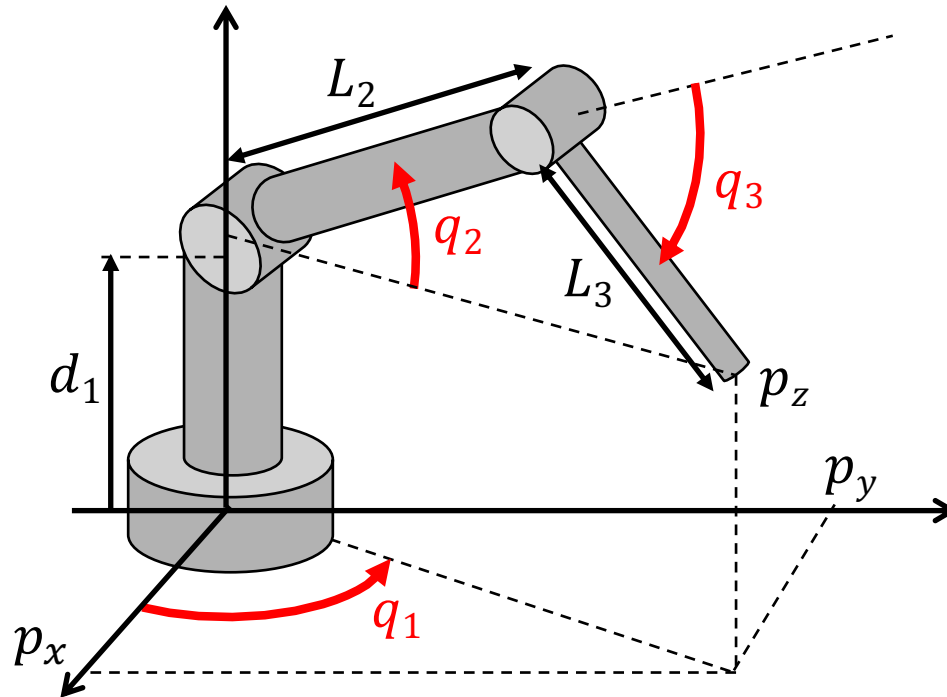
only when $p_x^2 + p_y^2 > 0$...
(else q_1 is **undefined** —infinite solutions!)

$$\Rightarrow \begin{cases} c_1 = p_x / \pm \sqrt{p_x^2 + p_y^2} \\ s_1 = p_y / \pm \sqrt{p_x^2 + p_y^2} \end{cases}$$

again, two solutions $\Rightarrow \begin{cases} q_1^{+} = \text{atan2}\{p_y, p_x\} \\ q_1^{-} = \text{atan2}\{-p_y, -p_x\} \end{cases}$



Inverse kinematics of 3R elbow-type arm step 3



combine first the two equations of direct kinematics and rearrange the last one

$$\begin{cases} c_1 p_x + s_1 p_y = L_2 c_2 + L_3 c_{23} \\ \quad \quad \quad = (L_2 + L_3 c_3) c_2 - L_3 s_3 s_2 \\ p_z - d_1 = L_2 s_2 + L_3 s_{23} \\ \quad \quad \quad = L_3 s_3 c_2 + (L_2 + L_3 c_3) s_2 \end{cases}$$

define and solve a **linear system** $Ax = b$
in the **algebraic** unknowns $x = (c_2, s_2)$

$$\begin{bmatrix} L_2 + L_3 c_3 & -L_3 s_3^{\{+,-\}} \\ L_3 s_3^{\{+,-\}} & L_2 + L_3 c_3 \end{bmatrix} \begin{bmatrix} c_2 \\ s_2 \end{bmatrix} = \begin{bmatrix} c_1^{\{+,-\}} p_x + s_1^{\{+,-\}} p_y \\ p_z - d_1 \end{bmatrix}$$

coefficient matrix A

known vector b

provided $\det A = p_x^2 + p_y^2 + (p_z - d_1)^2 \neq 0$

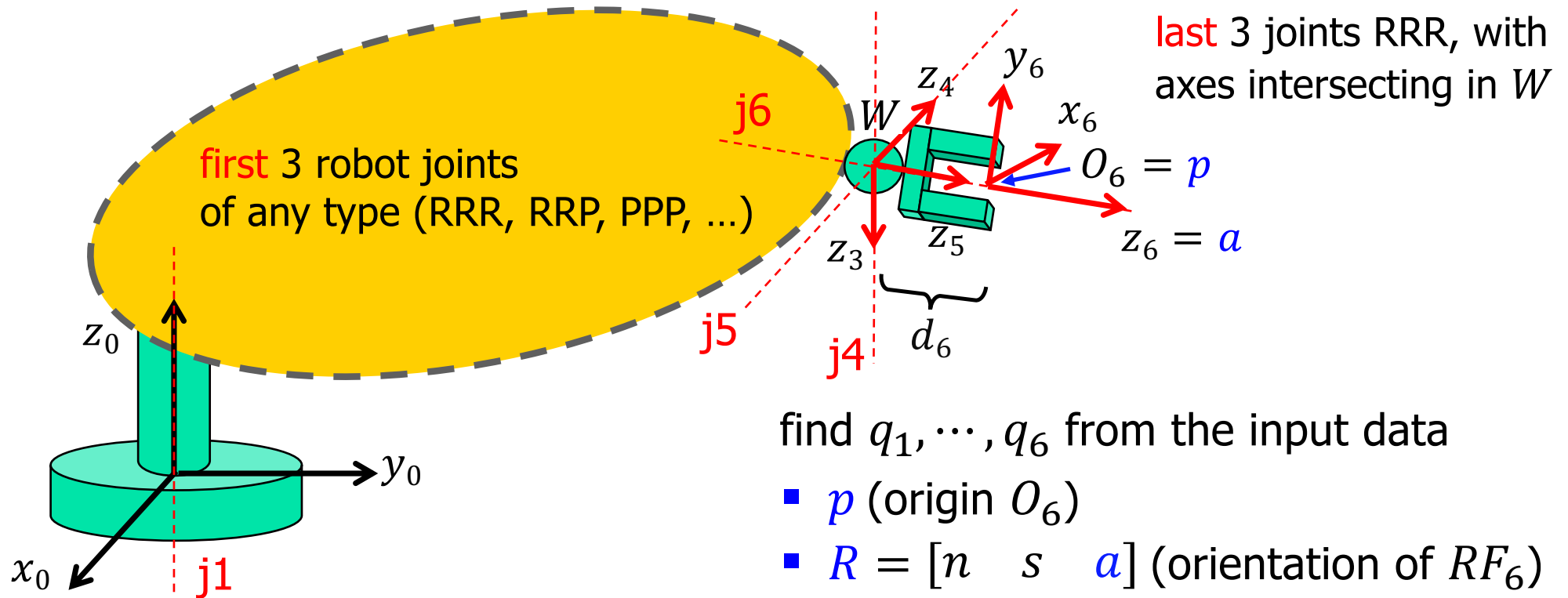
(else q_2 is **undefined** —infinite solutions!)

➔ 4 **regular** solutions for q_2 ,
depending on the combinations
of $\{+, -\}$ from q_1 and q_3

$$q_2^{\{\{f,b\},\{u,d\}\}} = \text{atan2} \left\{ s_2^{\{\{f,b\},\{u,d\}\}}, c_2^{\{\{f,b\},\{u,d\}\}} \right\}$$



Inverse kinematics for robots with spherical wrist

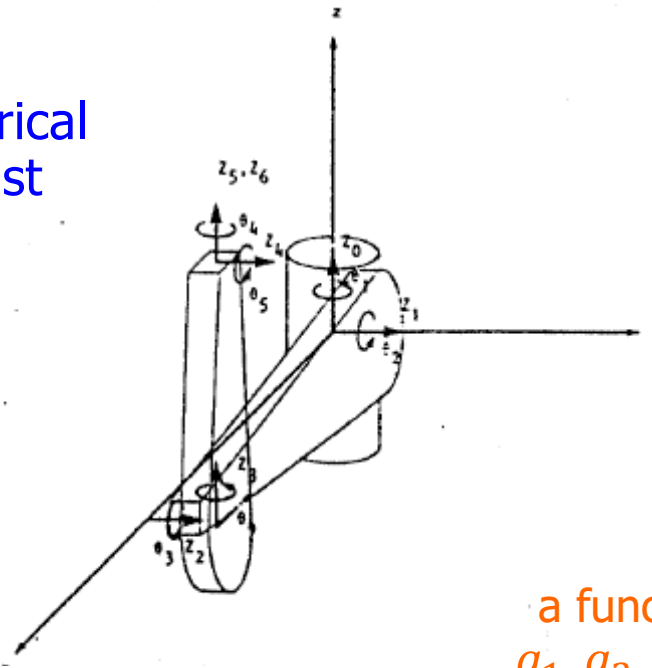


1. $W = p - d_6 a \Rightarrow q_1, q_2, q_3$ (inverse "position" kinematics for main axes)
 2. $R = {}^0R_3(q_1, q_2, q_3) \underbrace{{}^3R_6(q_4, q_5, q_6)}_{\text{Euler ZYZ or ZXZ rotation matrix with } q_4, q_5, q_6 (\theta_4, \theta_5, \theta_6)} \Rightarrow {}^3R_6(q_4, q_5, q_6) = {}^0R_3^T R \Rightarrow q_4, q_5, q_6$ (inverse "orientation" kinematics for the wrist)
- given known, after step 1 two regular solutions



6R robot Unimation PUMA 600

spherical wrist



a function of q_1, q_2, q_3 only!

TABLE I
LINK PARAMETERS FOR PUMA ARM

Joint	α°	θ°	d	a	Range
1	-90°	θ_1	0	0	$\theta_1: +/ - 160^\circ$
2	0	θ_2	0	a_2	$\theta_2: +45 \rightarrow -225^\circ$
3	90°	θ_3	d_3	a_3	$\theta_3: 225^\circ \rightarrow -45^\circ$
4	-90°	θ_4	d_4	0	$\theta_4: +/ - 170^\circ$
5	90°	θ_5	0	0	$\theta_5: +/ - 135^\circ$
6	0	θ_6	0	0	$\theta_6: +/ - 170^\circ$

$a_2 = 17.000$ $a_3 = 0.75$
 $d_3 = 4.937$ $d_4 = 17.000$

here $d_6 = 0$,

so that $O_6 = W$ directly

$$\begin{aligned}
 \left. \begin{aligned}
 n_x &= C_1[C_{23}(C_4C_5C_6 - S_4S_6) - S_{23}S_5C_6] \\
 &\quad - S_1[S_4C_5C_6 + C_4S_6] \\
 n_y &= S_1[C_{23}(C_4C_5C_6 - S_4S_6) - S_{23}S_5C_6] \\
 &\quad + C_1[S_4C_5C_6 + C_4S_6] \\
 n_z &= -S_{23}(C_4C_5C_6 - S_4S_6) - C_{23}S_5C_6
 \end{aligned} \right\} n = {}^0x_6(q) \\
 \left. \begin{aligned}
 o_x &= C_1[-C_{23}(C_4C_5S_6 + S_4C_6) + S_{23}S_5S_6] \\
 &\quad - S_1[-S_4C_5S_6 + C_4C_6] \\
 o_y &= S_1[-C_{23}(C_4C_5S_6 + S_4C_6) + S_{23}S_5S_6] \\
 &\quad + C_1[-S_4C_5S_6 + C_4C_6] \\
 o_z &= S_{23}(C_4C_5S_6 + S_4C_6) + C_{23}S_5S_6
 \end{aligned} \right\} s = {}^0y_6(q) \\
 \left. \begin{aligned}
 a_x &= C_1(C_{23}C_4S_5 + S_{23}C_5) - S_1S_4S_5 \\
 a_y &= S_1(C_{23}C_4S_5 + S_{23}C_5) + C_1S_4S_5 \\
 a_z &= -S_{23}C_4S_5 + C_{23}C_5
 \end{aligned} \right\} a = {}^0z_6(q) \\
 \left. \begin{aligned}
 p_x &= C_1(d_4S_{23} + a_3C_{23} + a_2C_2) - S_1d_3 \\
 p_y &= S_1(d_4S_{23} + a_3C_{23} + a_2C_2) + C_1d_3 \\
 p_z &= -(-d_4C_{23} + a_3S_{23} + a_2S_2)
 \end{aligned} \right\} p = {}^0o_6(q)
 \end{aligned}$$

8 different (regular) inverse solutions
that can be found in closed form

Finding nice kinematic relations

whiteboard ...



- the most complex inverse kinematics that can be solved in principle in closed form (i.e., **analytically**) is that of a **6R serial manipulator**, with arbitrary DH table
 - ways to systematically generate equations from the direct kinematics that could be easier to solve $\Rightarrow$ some scalar equations may contain perhaps **a single unknown variable!**

method used for the Unimation PUMA 600 in (*)

$${}^0T_6 = {}^0A_1(\theta_1) {}^1A_2(\theta_2) \cdots {}^5A_6(\theta_6) = U_0$$

$${}^0A_1^{-1} {}^0T_6 = U_1 (= {}^1A_2 \cdots {}^5A_6)$$

$${}^1A_2^{-1} {}^0A_1^{-1} {}^0T_6 = U_2 (= {}^2A_3 \cdots {}^5A_6)$$

$$\dots$$

$${}^4A_5^{-1} \cdots {}^1A_2^{-1} {}^0A_1^{-1} {}^0T_6 = U_5 (= {}^5A_6)$$

or also ...

$${}^0T_6 {}^5A_6^{-1} = V_5 (= {}^1A_2 \cdots {}^4A_5)$$

$${}^0T_6 {}^5A_6^{-1} {}^4A_5^{-1} = V_4 (= {}^1A_2 \cdots {}^3A_4)$$

$$\dots$$

$${}^0T_6 {}^5A_6^{-1} {}^4A_5^{-1} \cdots {}^1A_2^{-1} = V_1 (= {}^0A_1)$$

(*) Paul, Shimano, and Mayer: IEEE Transactions on Systems, Man, and Cybernetics, 1981

- generating from the direct kinematics a reduced set of equations to be solved (setting w.l.o.g. $d_1 = d_6 = 0$) $\Rightarrow$ **4 compact scalar equations** in the 4 unknowns $\theta_2, \dots, \theta_5$

$${}^0T_6 = \begin{bmatrix} n & s & a & p \\ 0 & 0 & 0 & 1 \end{bmatrix} = {}^0A_6(\theta) \longrightarrow \begin{aligned} a_z &= a^T(\theta) z & \|p\|^2 &= p^T(\theta) p(\theta) \\ p_z &= p^T(\theta) z & p^T a &= p^T(\theta) a(\theta) \end{aligned}$$

$z = [0 \quad 0 \quad 1]^T$

... then solve easily for the remaining θ_1 and θ_6

solved analytically or numerically ...

Manseur and Doty: International Journal of Robotics Research, 1988

Numerical solution of inverse kinematics problems



- use when a closed-form solution q to $r_d = f_r(q)$ does not exist or is “too hard” to be found
 - all methods are **iterative** and need the matrix $J_r(q) = \frac{\partial f_r(q)}{\partial q}$ (analytical Jacobian)
 - **Newton method** (here only for $m = n$, at the k th iteration)
 - $r_d = f_r(q) = f_r(q^k) + J_r(q^k)(q - q^k) + o(\|q - q^k\|)$ ← neglected
- $$q^{k+1} = q^k + J_r^{-1}(q^k) [r_d - f_r(q^k)]$$
- convergence for q^0 (initial guess) close enough to some $q^*: f_r(q^*) = r_d$
 - problems near **singularities** of the Jacobian matrix $J_r(q)$
 - in case of robot redundancy ($m < n$), use the pseudo-inverse $J_r^\#(q)$
 - has **quadratic** convergence rate when near to a solution (fast!)



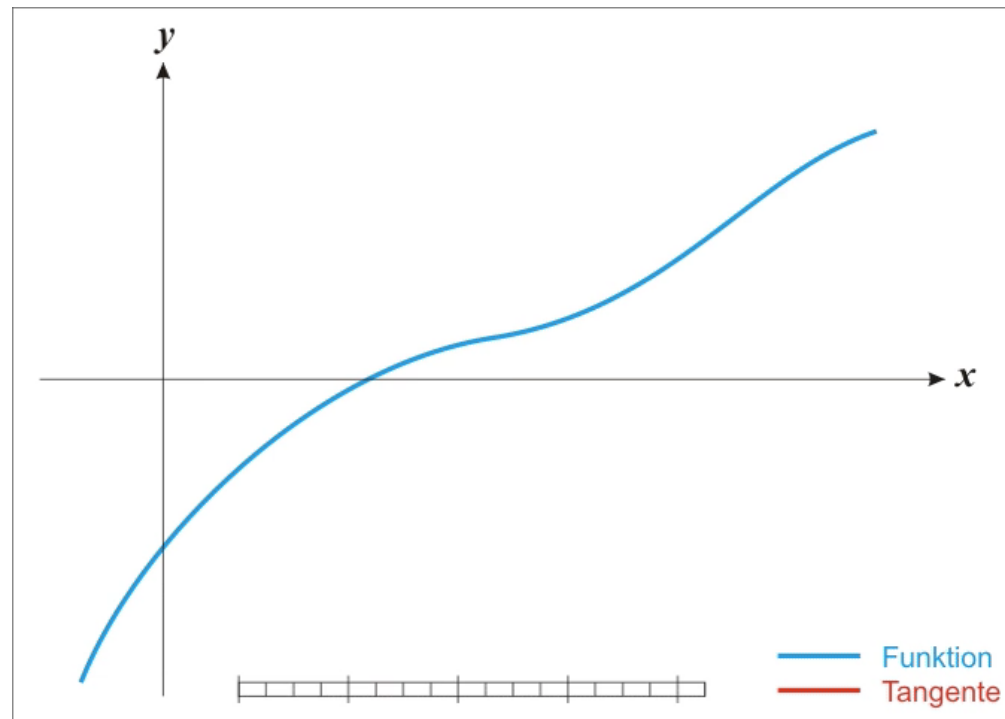
Operation of Newton method

- in the **scalar** case, also known as “method of the tangent”
- for a differentiable function $f(x)$, find a root x^* of $f(x^*) = 0$ by iterating as

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \quad \rightarrow$$

an approximating sequence

$$\{x_1, x_2, x_3, x_4, x_5, \dots\} \rightarrow x^*$$



animation from
http://en.wikipedia.org/wiki/File:NewtonIteration_Ani.gif

Numerical solution of inverse kinematics problems (cont'd)



- **Gradient method** (max descent)
 - minimize the **error** function

$$H(q) = \frac{1}{2} \|r_d - f_r(q)\|^2 = \frac{1}{2} (r_d - f_r(q))^T (r_d - f_r(q))$$

$$q^{k+1} = q^k - \alpha \nabla_q H(q^k)$$

from

$$\nabla_q H(q) = (\partial H(q) / \partial q)^T = - \left((r_d - f_r(q))^T (\partial f_r(q) / \partial q) \right)^T = -J_r^T(q) (r_d - f_r(q))$$

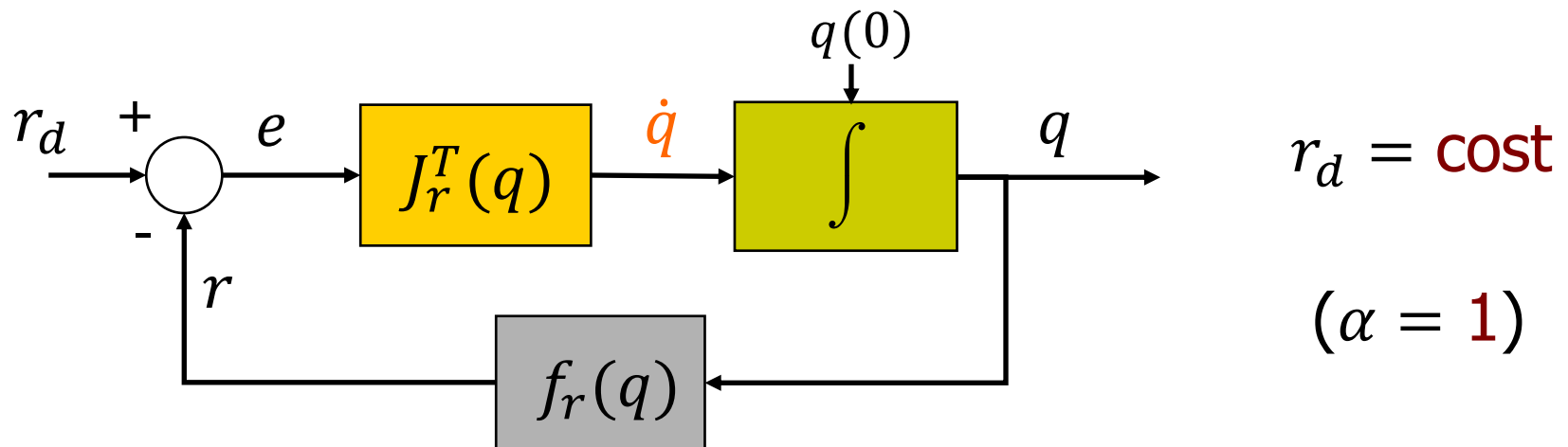
we get

$$q^{k+1} = q^k + \alpha J_r^T(q^k) (r_d - f_r(q^k))$$

- the scalar **step size** $\alpha > 0$ should be chosen so as to guarantee a decrease of the error function at each iteration: too large values for α may lead the method to “miss” the minimum
- when the step size is too small, convergence is extremely **slow**



Revisited as a feedback scheme



$e = r_d - f_r(q) \rightarrow 0 \iff$ closed-loop **equilibrium** $e = 0$
is **asymptotically stable**

$V = \frac{1}{2} e^T e \geq 0$ is a **Lyapunov** candidate function

$$\dot{V} = e^T \dot{e} = e^T \frac{d}{dt} (r_d - f_r(q)) = -e^T J_r(q) \dot{q} = -e^T J_r(q) J_r^T(q) e \leq 0$$

$\dot{V} = 0 \iff e \in \mathcal{N}(J_r^T(q))$ in particular, $e = 0$

↑
null space

asymptotic stability



Properties of Gradient method

- **computationally simpler**: use the **Jacobian transpose**, rather than its (pseudo)-inverse
- same use also for robots that are **redundant** ($n > m$) for the task
- may not converge to a solution, but it **never diverges**
- the **discrete-time** evolution of the continuous scheme

$$q^{k+1} = q^k + \Delta T J_r^T(q^k)(r_d - f_r(q^k)), \quad \alpha = \Delta T$$

is equivalent to an iteration of the Gradient method

- the scheme can be accelerated by using a gain matrix $K > 0$

$$\dot{q} = J_r^T(q) K e = J_r^T(q) K (r_d - f_r(q))$$

note: $K \rightarrow K + K_s$, with K_s skew-symmetric, can be used also to “escape” from being stuck in a **stationary point** of $V = \frac{1}{2} e^T K e$, by **rotating** the error Ke out of the null space of J_r^T (when a **singularity** is encountered)



A case study

analytic expressions of Newton and gradient iterations

- 2R robot with $l_1 = l_2 = 1$, desired end-effector position $r_d = p_d = (1,1)$
- direct kinematic function and error

$$f_r(q) = \begin{pmatrix} c_1 + c_{12} \\ s_1 + s_{12} \end{pmatrix} \quad e = p_d - f_r(q) = \begin{pmatrix} 1 \\ 1 \end{pmatrix} - f_r(q)$$

- Jacobian matrix

$$J_r(q) = \frac{\partial f_r(q)}{\partial q} = \begin{pmatrix} -(s_1 + s_{12}) & -s_{12} \\ c_1 + c_{12} & c_{12} \end{pmatrix}$$

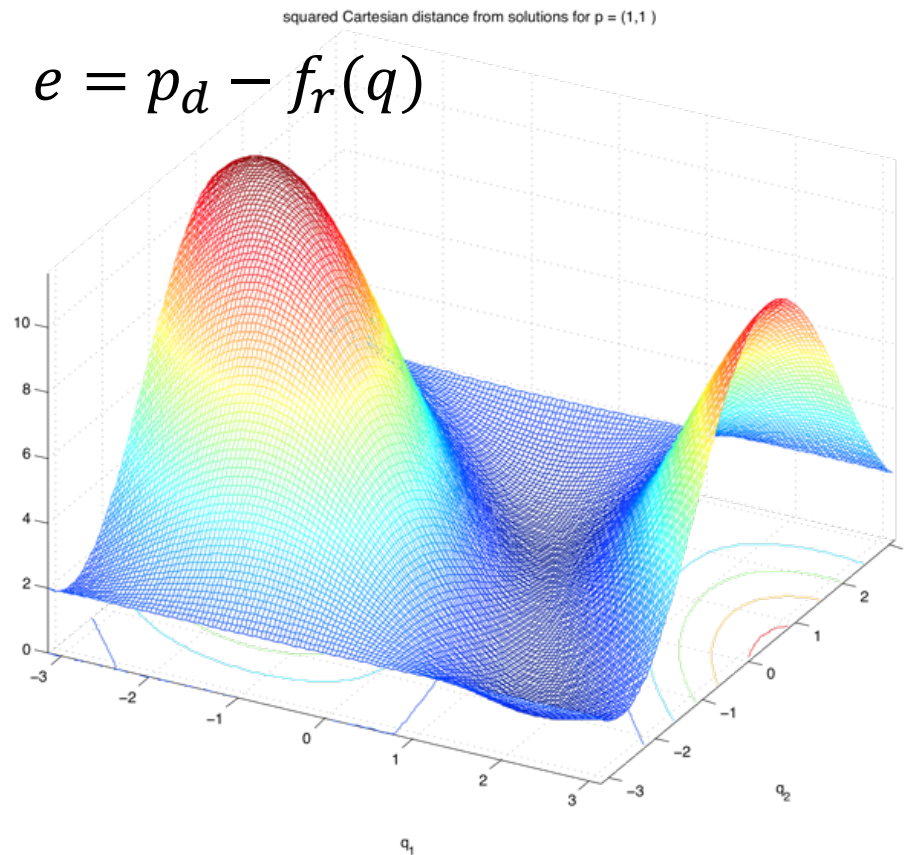
- **Newton** versus **Gradient** iteration

$$q^{k+1} = q^k + \underbrace{\det J_r(q)}_{\text{det } J_r(q)} \underbrace{J_r^{-1}(q^k)}_{J_r^{-1}(q^k)} \underbrace{e_k}_{e_k} \times \underbrace{J_r^T(q^k)}_{J_r^T(q^k)}$$

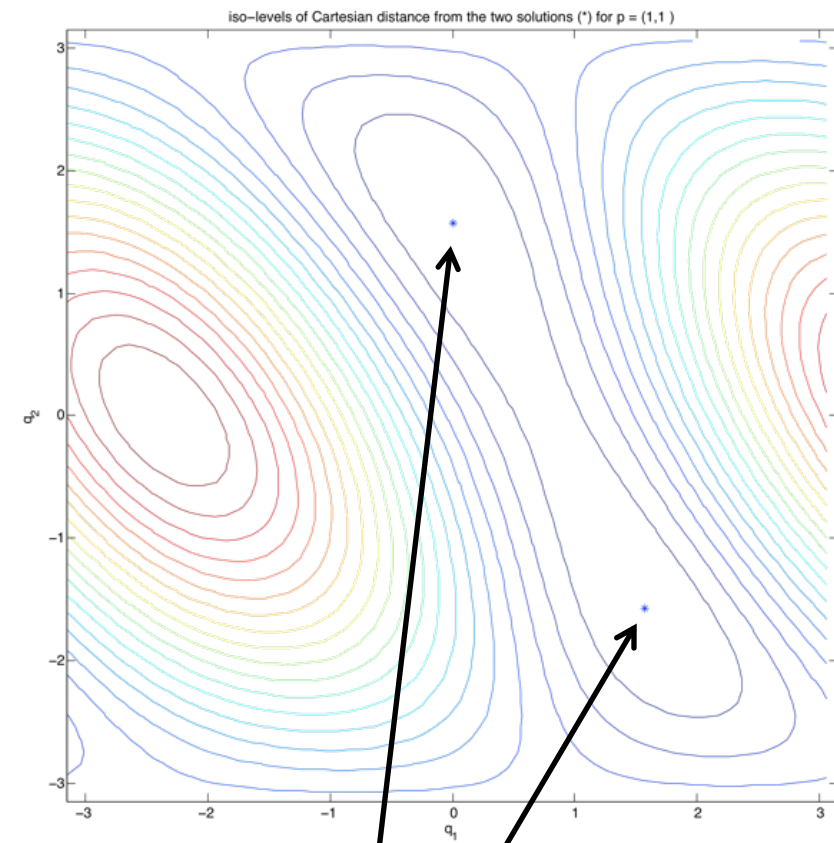
The equation shows the Newton iteration formula. A red arrow points from the text "det $J_r(q)$ " to the $\frac{1}{s_2}$ term in the matrix. The matrix is evaluated at $q = q^k$. The error vector e_k is shown in a green box. The Jacobian matrix $J_r^T(q^k)$ is shown in an orange box.

Error function

- 2R robot with $l_1 = l_2 = 1$ and desired end-effector position $p_d = (1,1)$



plot of $\|e\|^2$ as a function of $q = (q_1, q_2)$



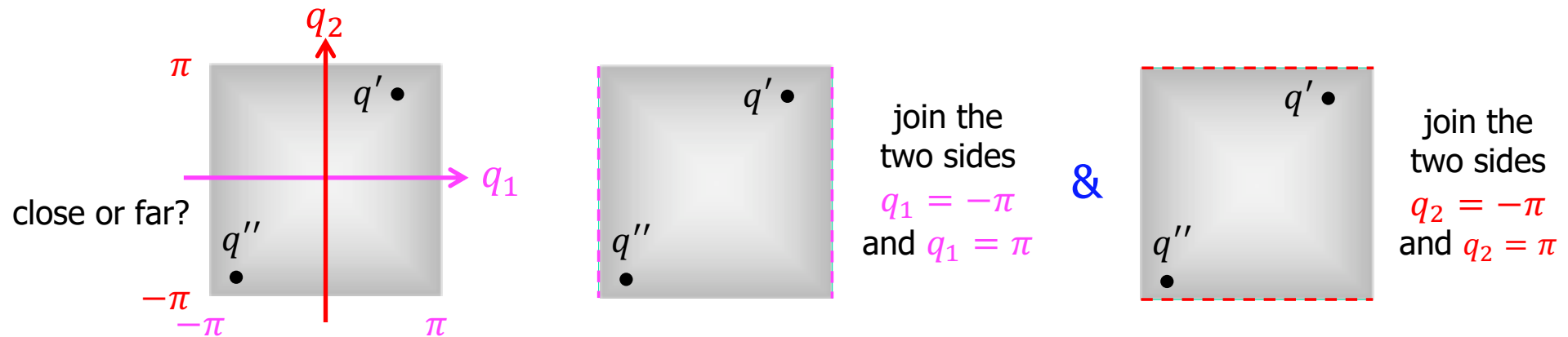
two local minima
(inverse kinematic solutions)



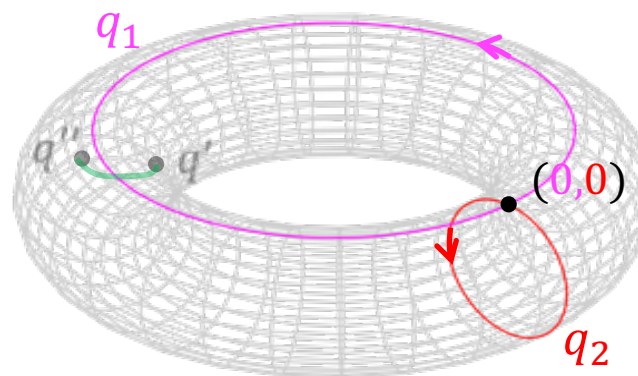
Configuration space of 2R robot

whiteboard ...

- can we represent the correct “distance” between two configurations q' and q'' of this robot on a (square) region in $\mathbb{R}^2$?



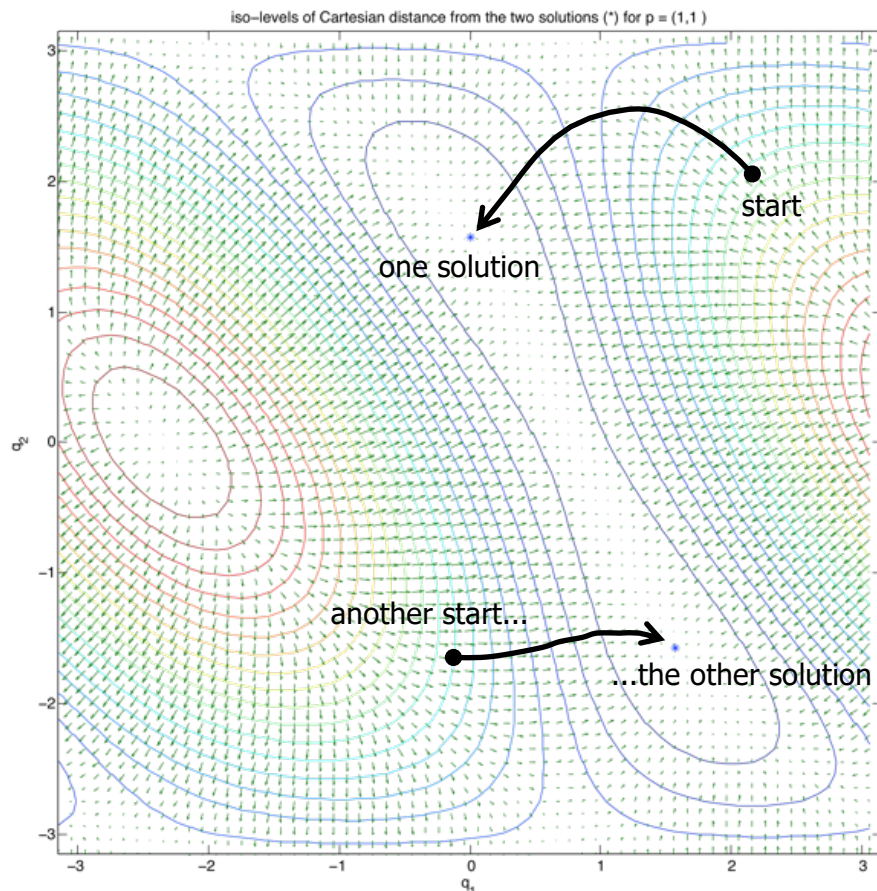
- configuration space is a **torus** $SO(1) \times SO(1)$, i.e., the surface of a “donut”



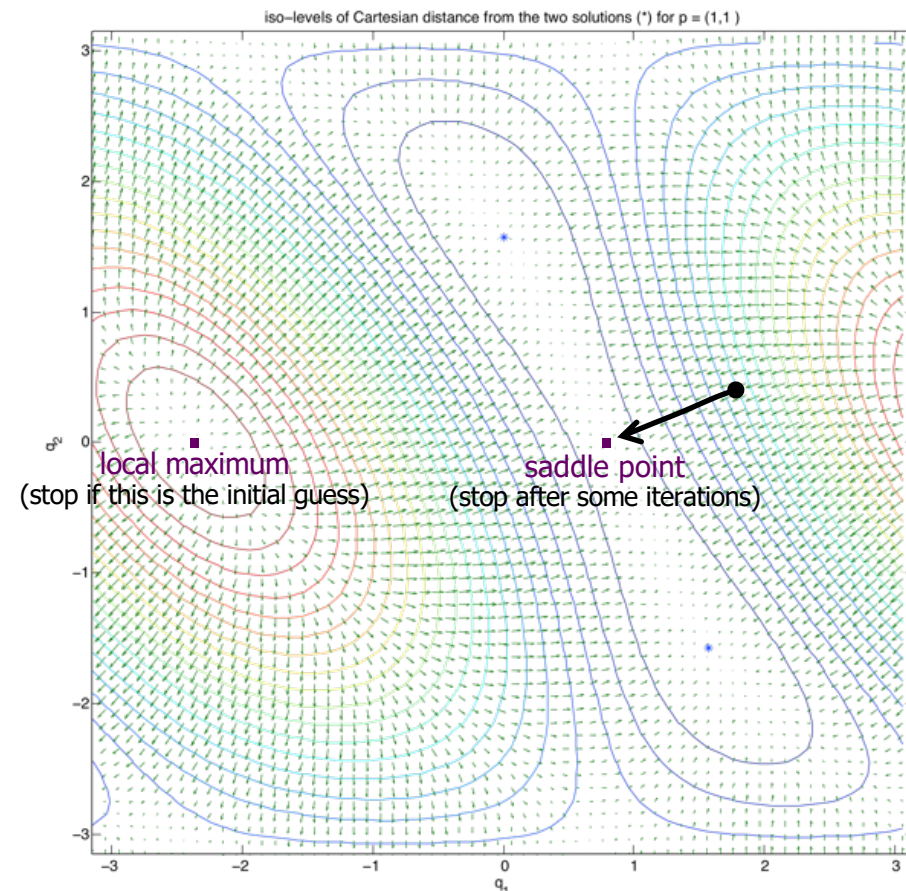
- the right metric is a **geodesic** on the torus ...

Error reduction by Gradient method

- flow of iterations along the **negative** (or anti-) gradient
- two possible cases: convergence or stuck (at **zero gradient**)



$$(q_1, q_2)' = (0, \pi/2) \quad (q_1, q_2)'' = (\pi/2, -\pi/2)$$



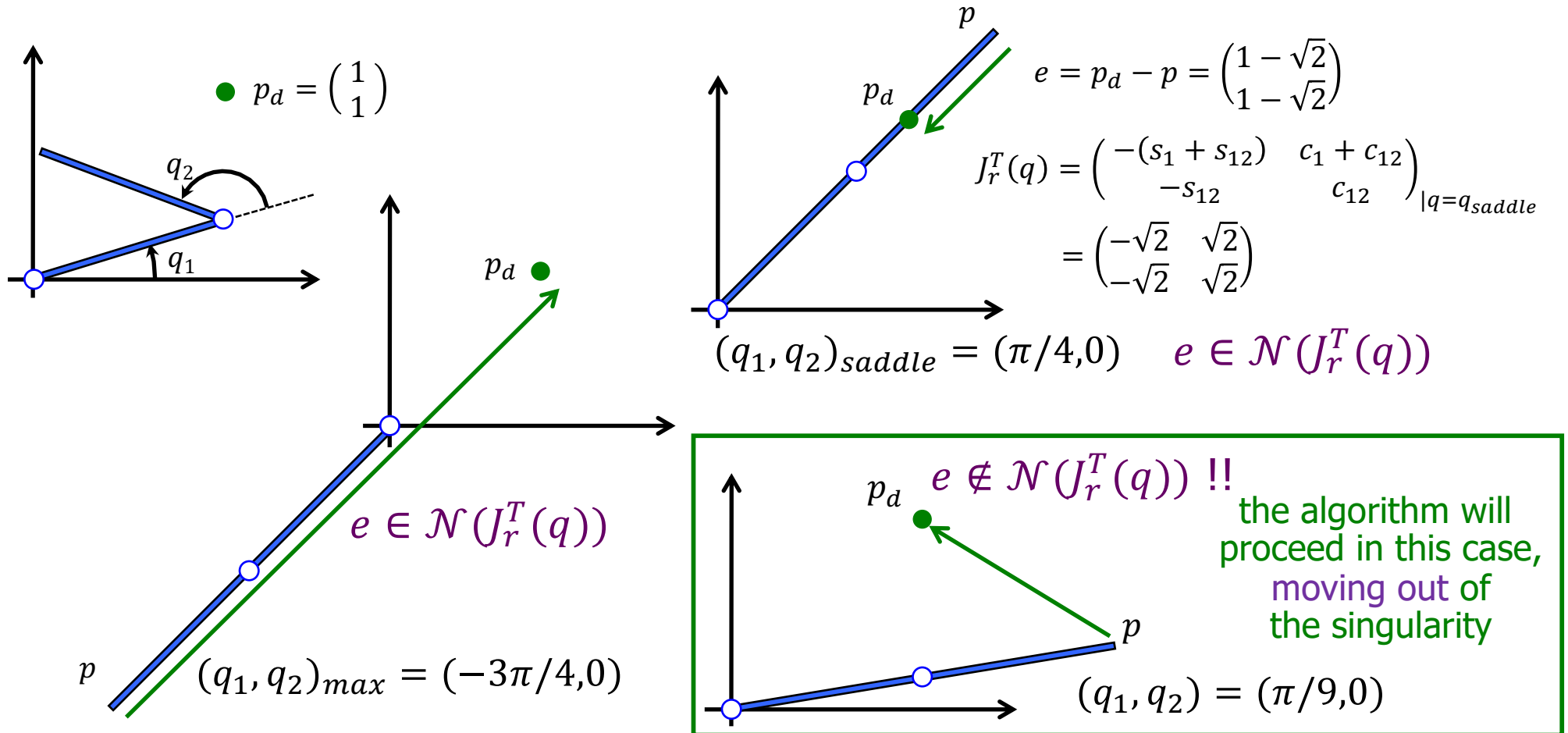
$$(q_1, q_2)_{max} = (-3\pi/4, 0) \quad (q_1, q_2)_{saddle} = (\pi/4, 0)$$

$$e \in \mathcal{N}(J_r^T(q)) !$$

Convergence analysis

when does the gradient method get stuck?

- lack of convergence occurs when
 - the Jacobian matrix $J_r(q)$ is not full rank (the robot is in a “singular configuration”)
 - **AND** the error e is in the null space of $J_r^T(q)$





Issues in implementation

- initial guess q^0
 - only **one** inverse solution is generated for each guess
 - multiple initializations for obtaining other solutions
- optimal step size $\alpha > 0$ in Gradient method
 - a constant step may work good initially, but not close to the solution (or vice versa)
 - an **adaptive** one-dimensional line search (e.g., Armijo's rule) could be used to choose the best α at each iteration

- stopping criteria

Cartesian error
(possibly, separate for position and orientation) $\|r_d - f_r(q^k)\| \leq \varepsilon$ **algorithm increment** $\|q^{k+1} - q^k\| \leq \varepsilon_q$

- understanding closeness to singularities

$$\sigma_{\min}\{J_r(q^k)\} \geq \sigma_0$$

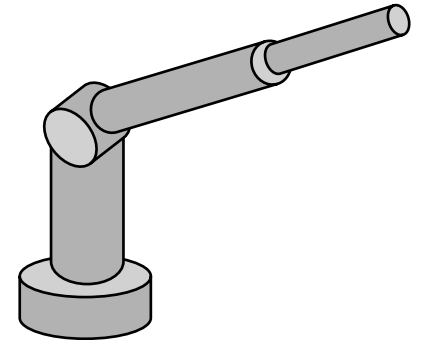
**good numerical conditioning
of Jacobian matrix (SVD)**

(or a simpler test on its determinant, for $m = n$)



Numerical tests on RRP robot

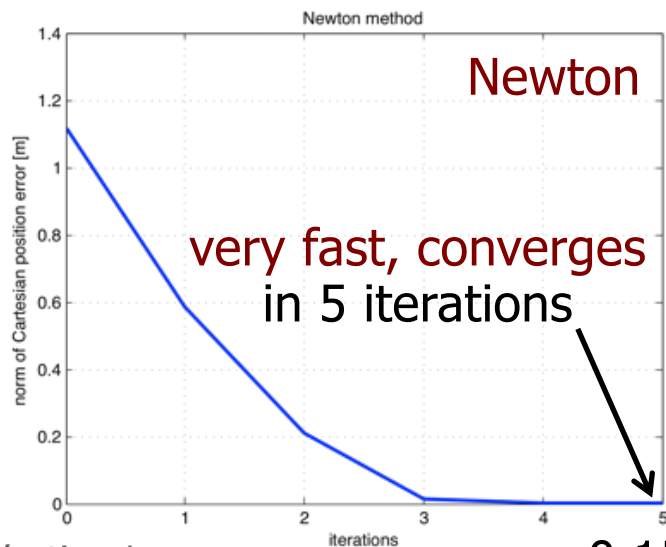
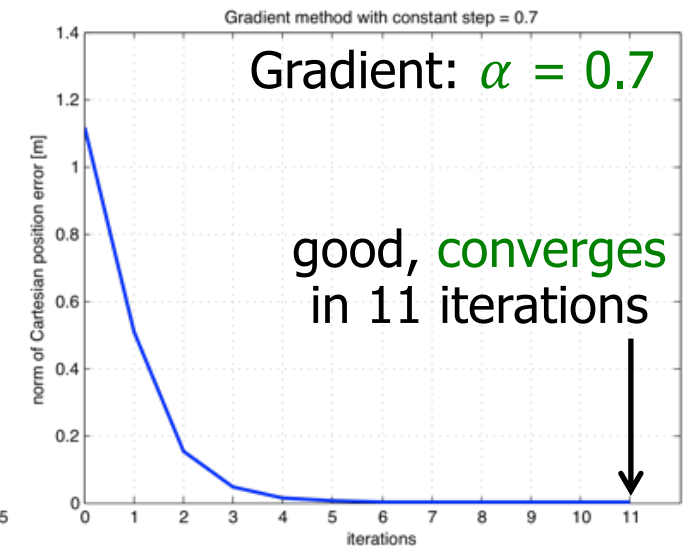
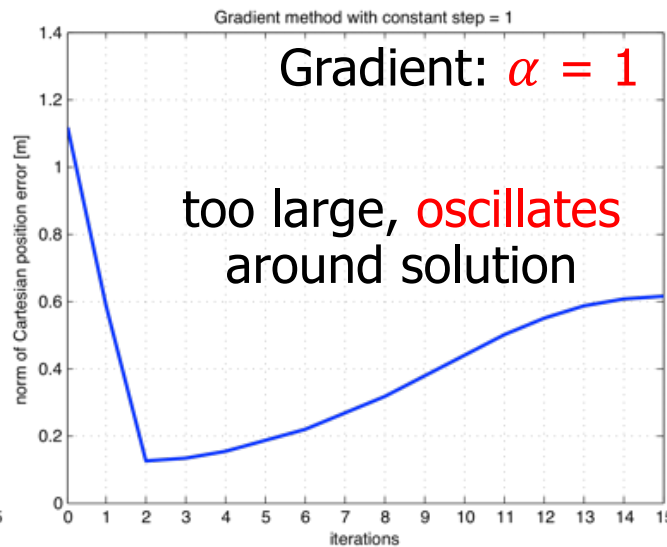
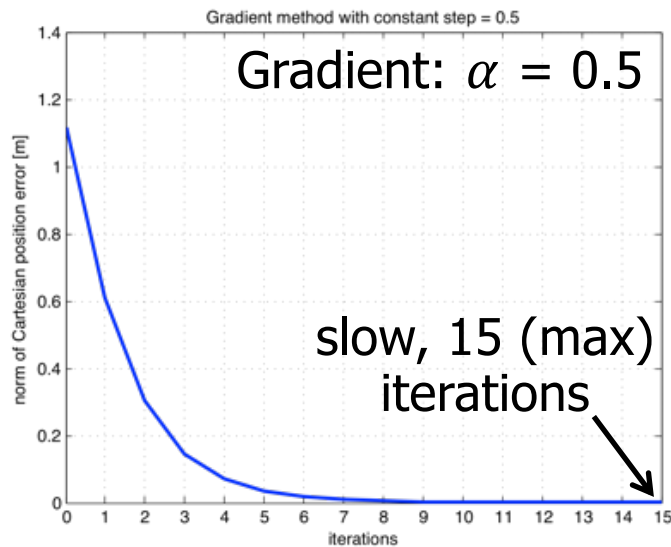
- **RRP/polar robot**: desired E-E position $r_d = p_d = (1, 1, 1)$
—see **slide 21**, with $d_1 = 0.5$
- the two (known) **analytical** solutions, with $q_3 \geq 0$, are
$$q^* = (0.7854, 0.3398, 1.5)$$
$$q^{**} = (q_1^* - \pi, \pi - q_2^*, q_3^*) = (-2.3562, 2.8018, 1.5)$$
- norms $\varepsilon = 10^{-5}$ (max Cartesian error), $\varepsilon_q = 10^{-6}$ (min joint increment)
- $k_{max} = 15$ (max # iterations), $|\det J_r(q)| \leq 10^{-4}$ (singularity closeness)
- **numerical** performance of Gradient (with different steps α) vs. Newton
 - **test 1**: $q^0 = (0, 0, 1)$ as initial guess
 - **test 2**: $q^0 = (-\pi/4, \pi/2, 1)$ — “singular” start, since $c_2 = 0$ (see **slide 21**)
 - **test 3**: $q^0 = (0, \pi/2, 0)$ — “doubly singular” start, since also $q_3 = 0$
 - solution and plots with Matlab code



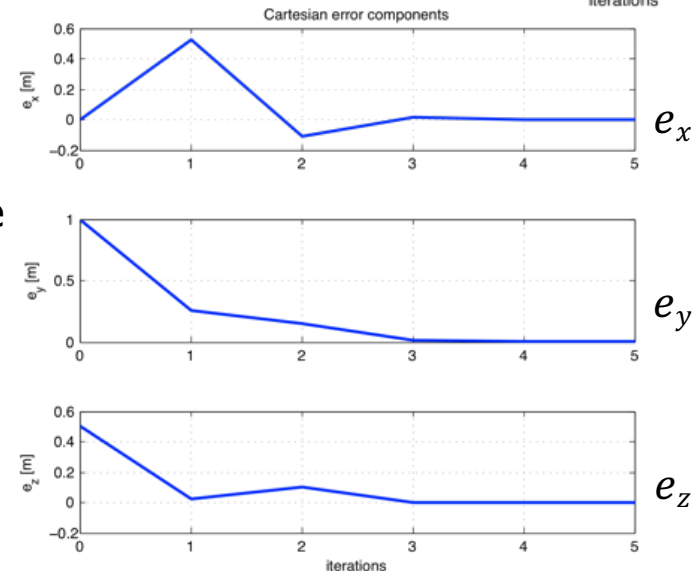


Numerical test - 1

- test 1: $q^0 = (0, 0, 1)$ as initial guess; evolution of error norm



Cartesian errors component-wise

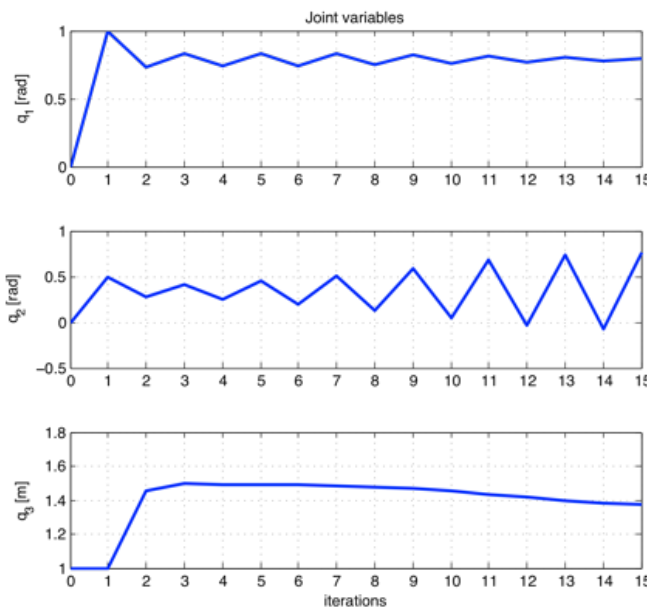


$0.57 \cdot 10^{-5}$

$0.15 \cdot 10^{-8}$

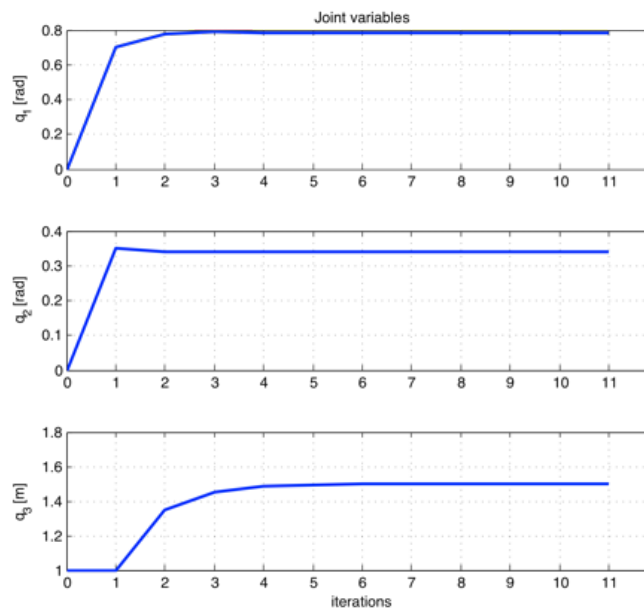
Numerical test - 1

- test 1: $q^0 = (0, 0, 1)$ as initial guess; evolution of joint variables



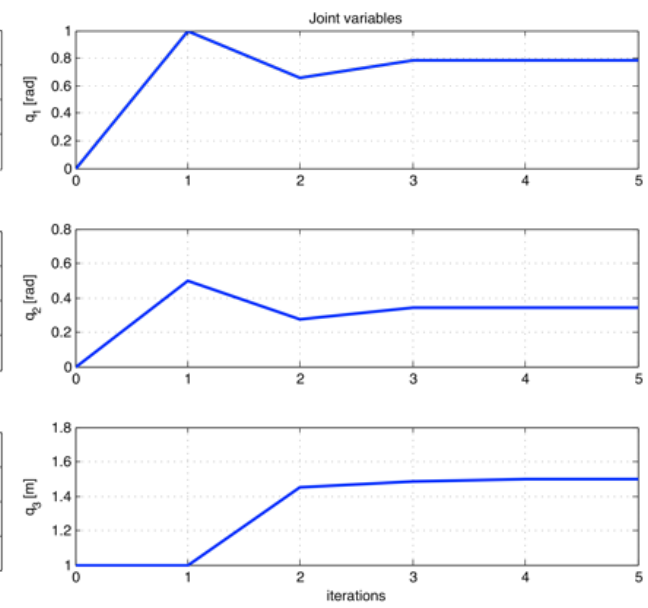
Gradient: $\alpha = 1$

not converging
to a solution



Gradient: $\alpha = 0.7$

converges in
11 iterations



Newton

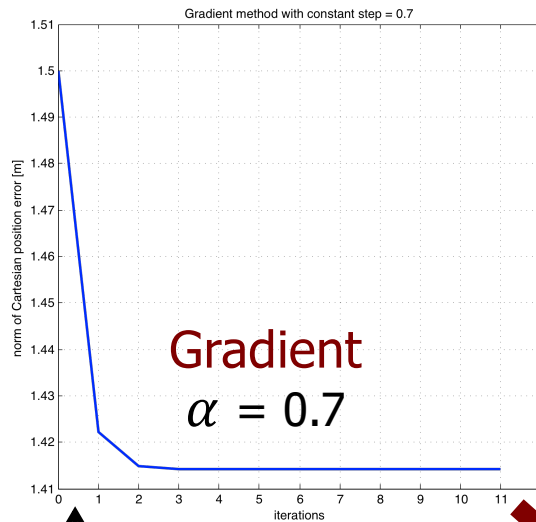
converges in
5 iterations

both to the same solution $q^* = (0.7854, 0.3398, 1.5)$

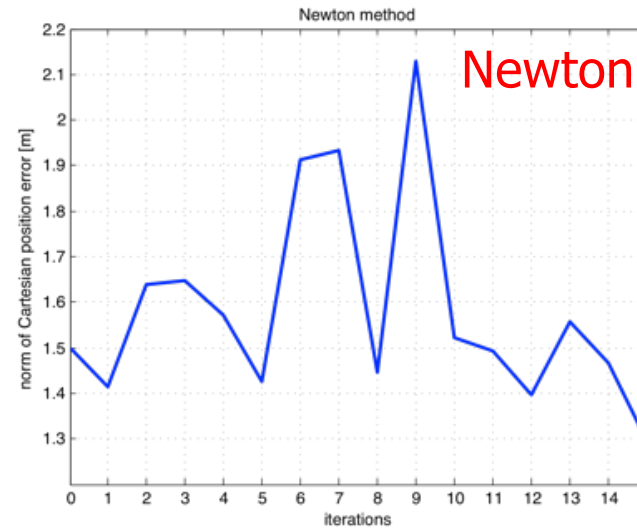
Numerical test - 2

- test 2: $q^0 = (-\pi/4, \pi/2, 1)$: singular start

error norms



Gradient
 $\alpha = 0.7$



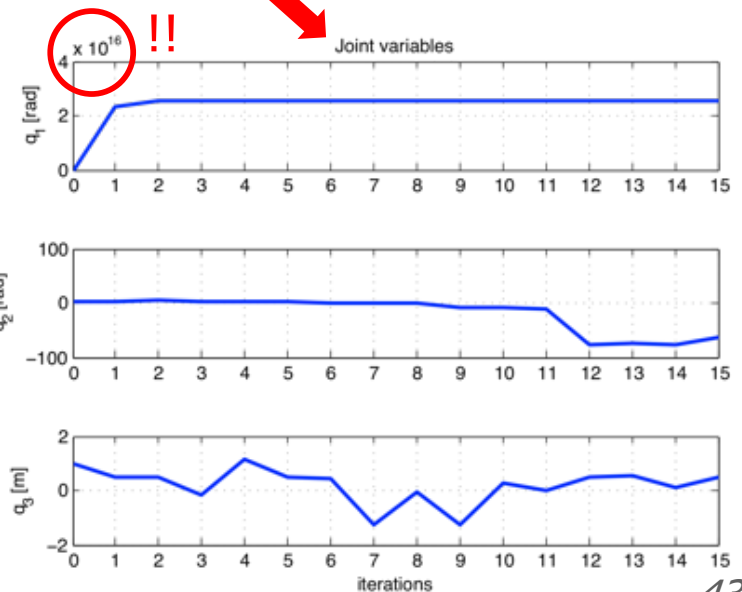
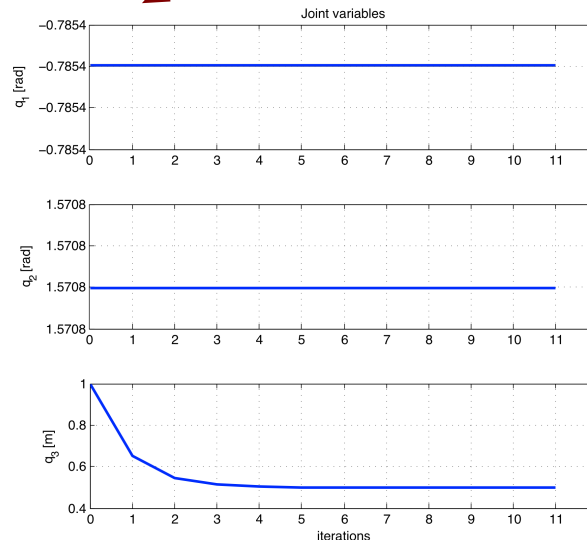
Newton

with check of singularity:
blocked at start

without check:
it diverges!

starts toward solution, but slowly stops (in singularity): when Cartesian error vector $e \in \mathcal{N}(J_r^T(q))$

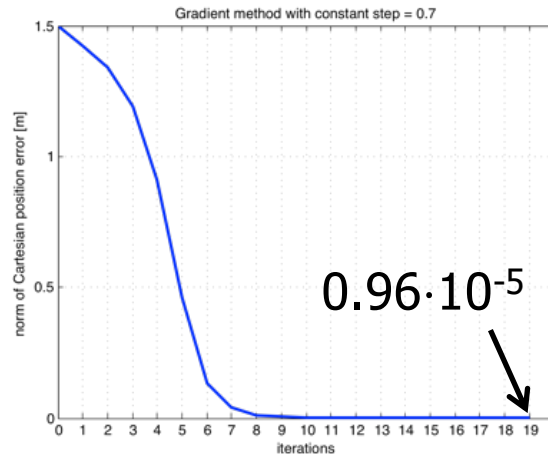
joint variables



Numerical test - 3

- test 3: $q^0 = (-\pi/4, \pi/2, 1)$: doubly singular start

error norm



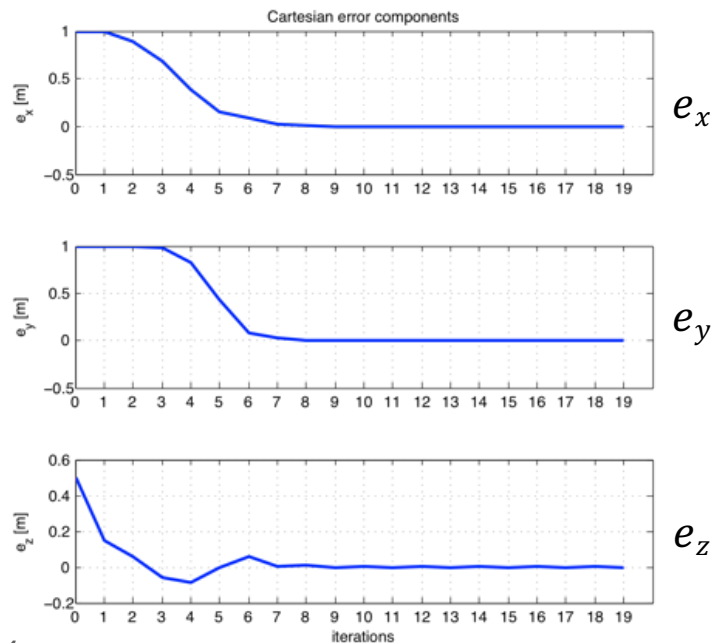
Gradient (with $\alpha = 0.7$)

- ① starts toward solution
- ② exits the double singularity
- ③ slowly converges in 19 iterations to the solution

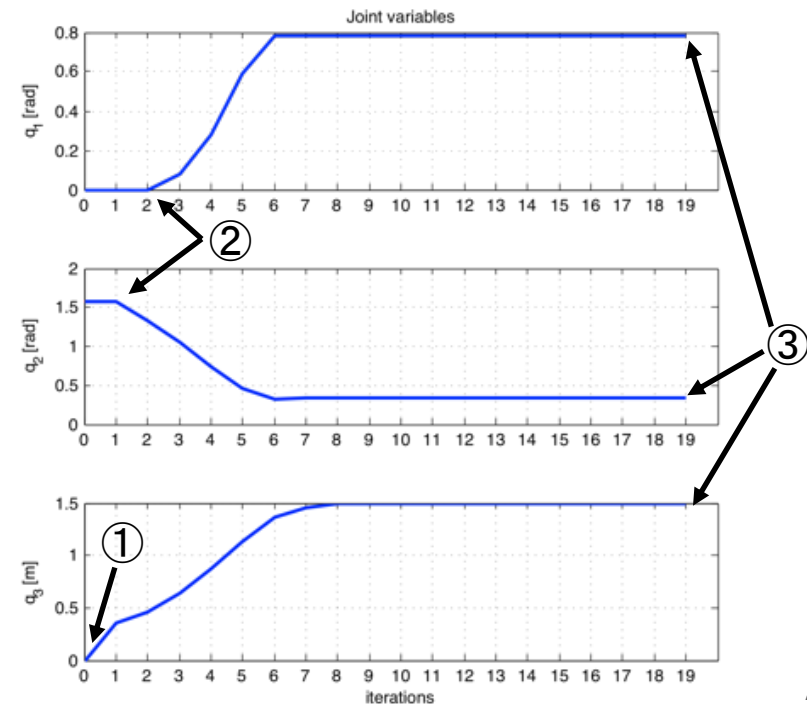
$$q^* = (0.7854, 0.3398, 1.5)$$

Newton is either blocked at start or (w/o check) explodes! ⇒ "NaN" in Matlab

Cartesian errors



joint variables





Final remarks

- an **efficient** iterative scheme can be devised by combining
 - **initial iterations** using Gradient (“sure but slow”, linear convergence rate)
 - **switch then** to Newton method (quadratic terminal convergence rate)
- **joint range limits** are considered only at the end
 - check if the solution found is **feasible**, as for analytical methods
- in alternative, an **optimization** criterion can be included in the search
 - drives iterations toward an inverse kinematic solution with nicer properties
- if the problem has to be solved **on-line**
 - execute iterations and associate an actual robot motion: **repeat steps** at times $t_0, t_1 = t_0 + T, \dots, t_k = t_{k-1} + T$ (e.g., every $T = 40$ ms)
 - a “good” choice for the initial guess q^0 at t_k is the solution of the previous problem at t_{k-1} (provides continuity, requires only 1-2 Newton iterations)
 - crossing of singularities and handling of joint range limits need special care
- Jacobian-based inversion schemes are used also for **kinematic control**, moving along a continuous task trajectory $r_d(t)$